



**UNIVERSIDAD NACIONAL DE INGENIERÍA**

**Facultad de Ingeniería  
Industrial y de Sistemas**

*Estándares de Ingeniería de Sistemas – SI-705  
Metodologías Agiles parte*

# El contexto de los proyectos

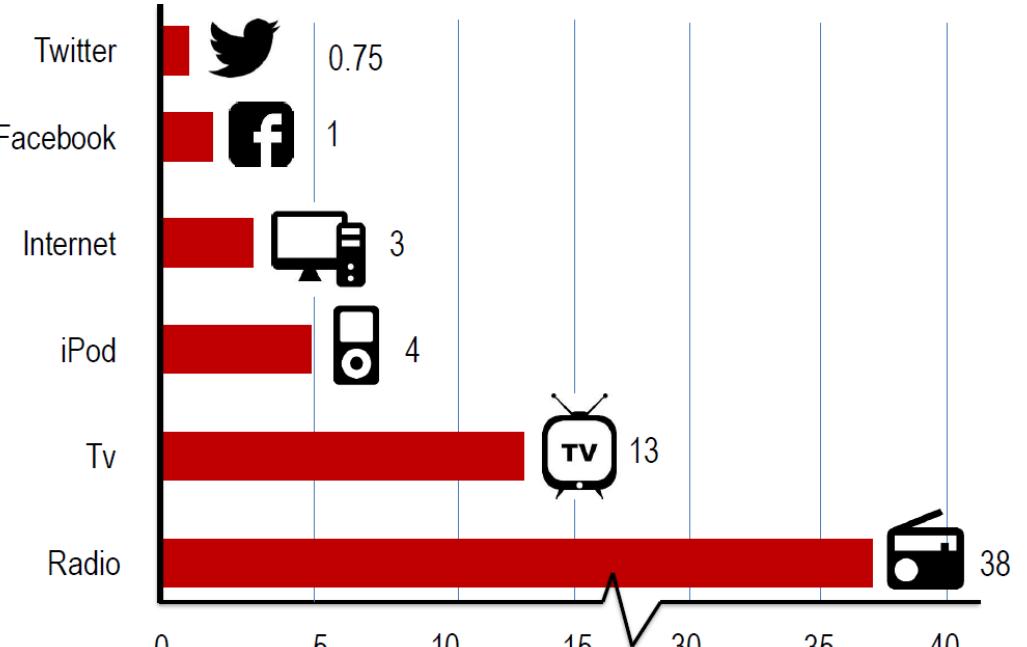
## Marco Cynefin

5 Dominios para explicar la relación entre el grado de incertidumbre y

## Entorno VUCA

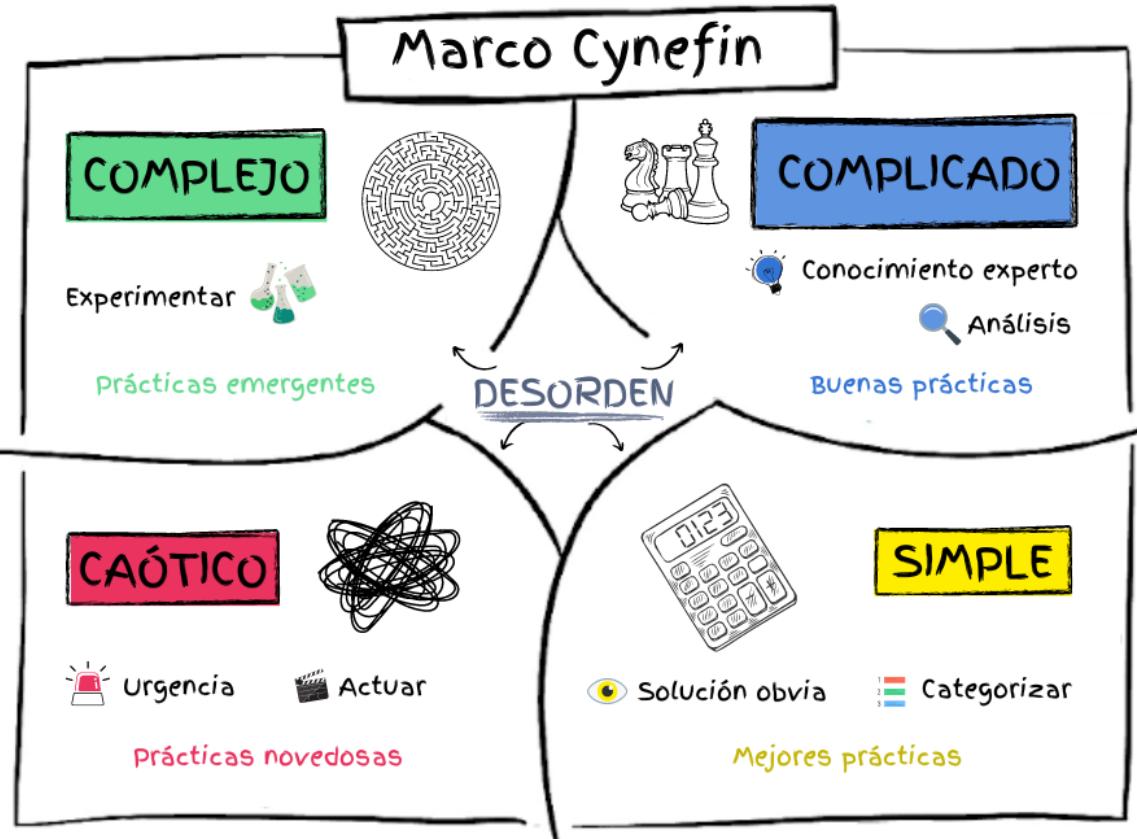
Que explica las características de un contexto cambiante

Tiempo en años para llegar a 50 millones de usuarios



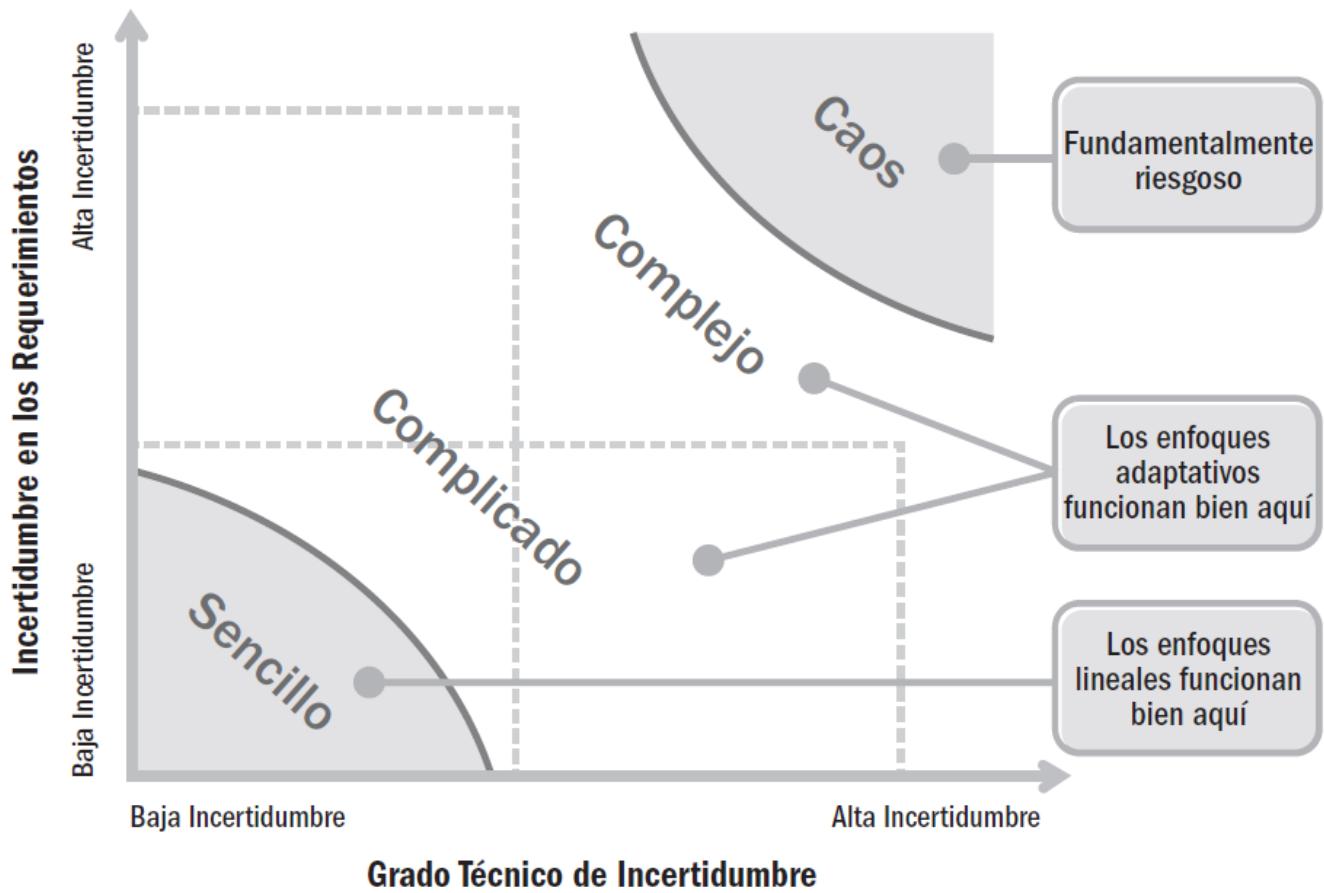
Fuente: Statista

# El contexto de los proyectos



	Relacion Causa-Efecto	Preguntas Clave
<b>Simple</b>	Es obvia para todos	¿Podemos aplicar una regla o una mejor práctica conocida? ¿La situación se entiende y es predecible?
<b>Complicado</b>	Requiere análisis y/o conocimientos expertos.	¿Necesitamos expertos para analizar o descomponer el problema? ¿Hay varias formas correctas de abordar esta situación?
<b>Complejo</b>	Solo se puede determinar en retrospectiva.	¿La situación es predecible? ¿Necesitamos experimentar?
<b>Caótico</b>	No se ven posibles soluciones.	¿Se requiere una acción inmediata? ¿Las circunstancias son tan turbulentas que cualquier acción es mejor que ninguna?
<b>Desorden</b>	Estado de confusión.	¿Estamos confundidos sobre cómo proceder? ¿No tenemos clara la naturaleza del problema?

# Incertidumbre en el contexto



Fuente: Guía Práctica de Agilidad (PMI)

# Ejemplo

## 1. Dominio Caótico

Al principio, nadie sabía qué hacer para volar. Se sospechaba que sería posible, imitando a los pájaros, pero nadie sabía cómo.



## 3. Dominio Complicado

Pasaron los años, y el diseño de los aviones empezó a mejorar. Sin embargo, había un problema: *Los vuelos largos transoceánicos eran un desafío enorme.*



## 2. Dominio Complejo

Tan pronto como los hermanos Wright crearon el primer prototipo viable de avión, todo el mundo empezó a ver hacia dónde iban las cosas → 2 alas, un propulsor frontal, etc.



## 4. Dominio Simple o Claro

Hoy en día, se sabe a la perfección cómo diseñar un avión seguro, durable y rápido, que se pueda pilotar a cualquier sitio.

**Claro o Simple, no es lo mismo que Fácil.**



# El contexto de los proyectos VUCA

## (V) Volatility:

Enfrentan una gran cantidad de cambios en poco tiempo y a gran velocidad.

## (U) Uncertainty:

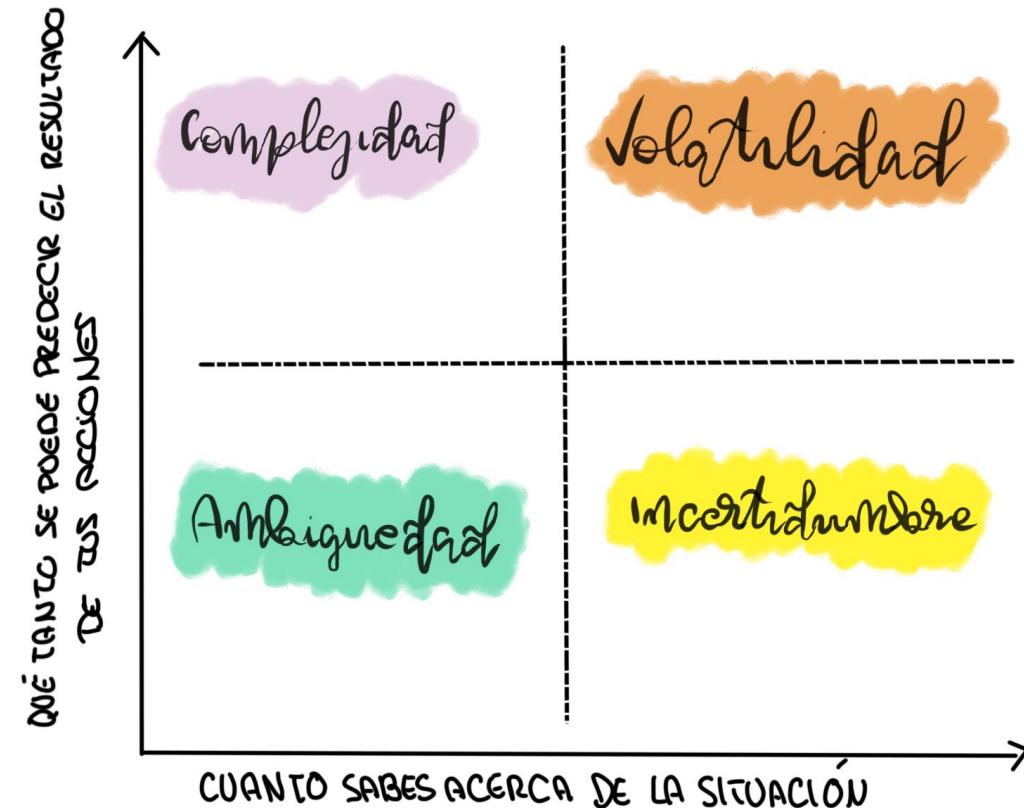
Tienen incapacidad para predecir las situaciones o el curso de los acontecimientos.

## (C) Complexity:

Tienen dificultad para entender y manejar el contexto y para distinguir entre causa y efecto.

## (A) Ambiguos:

Alude a la complejidad de entender las relaciones entre los elementos que están presentes en el entorno.



# Dinámica

¿Alguno de estos enfoques es el correcto y el otro erróneo?

1



2



# Ciclos de vida de proyectos

**Predictiva:** Un enfoque más tradicional, en el que la mayor parte de la planificación ocurre por adelantado, y luego se ejecuta en una sola pasada; es un proceso secuencial

**Iterativa:** Un enfoque que permite obtener retroalimentación para el trabajo sin terminar, a fin de mejorar y modificar ese trabajo

**Incremental:** Un enfoque que proporciona entregables terminados que el cliente puede utilizar de inmediato

**Adaptativa (Ágil)** : Un enfoque que es tanto iterativo como incremental a fin de refinar los elementos de trabajo y poder entregar con frecuencia

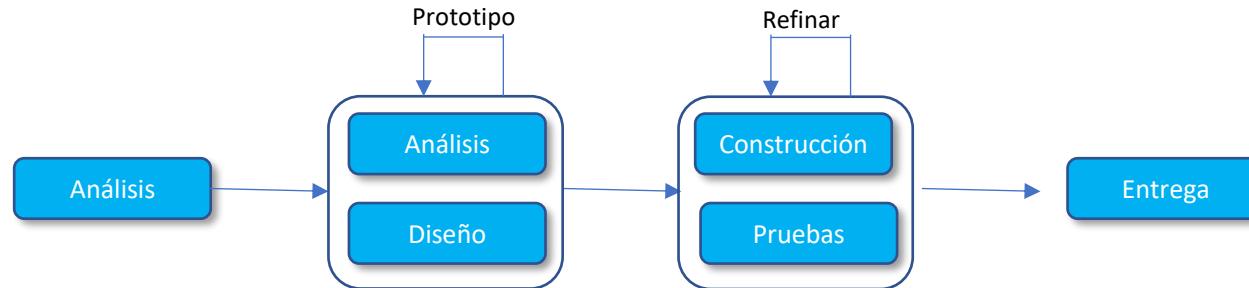


# Ciclos de vida de proyecto

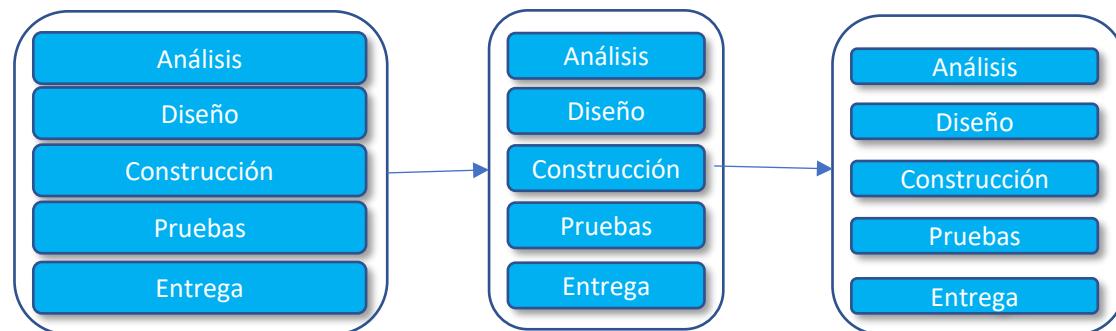
**Predictiva:**



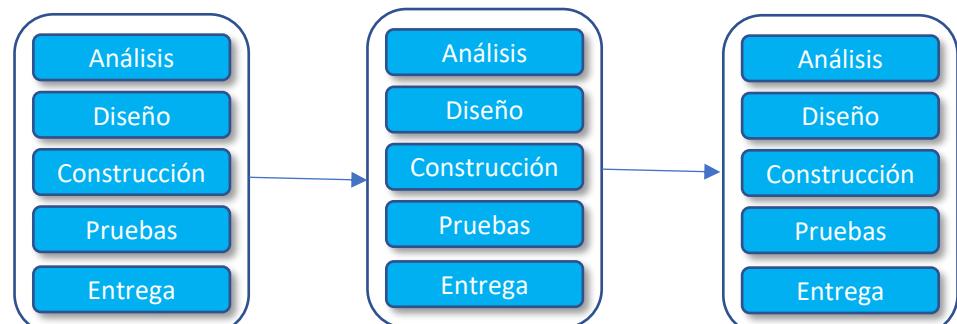
**Iterativa:**



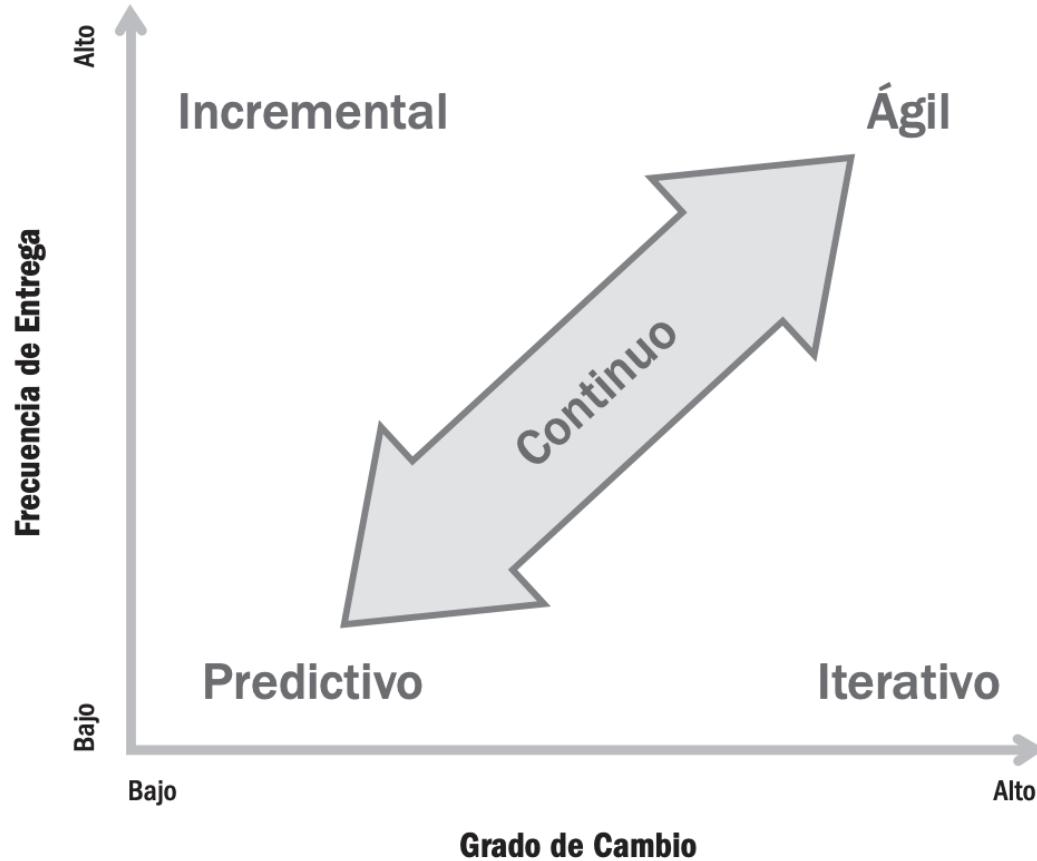
**Incremental:**



**Adaptativo:  
(Ágil)**



# Selección de ciclo de vida



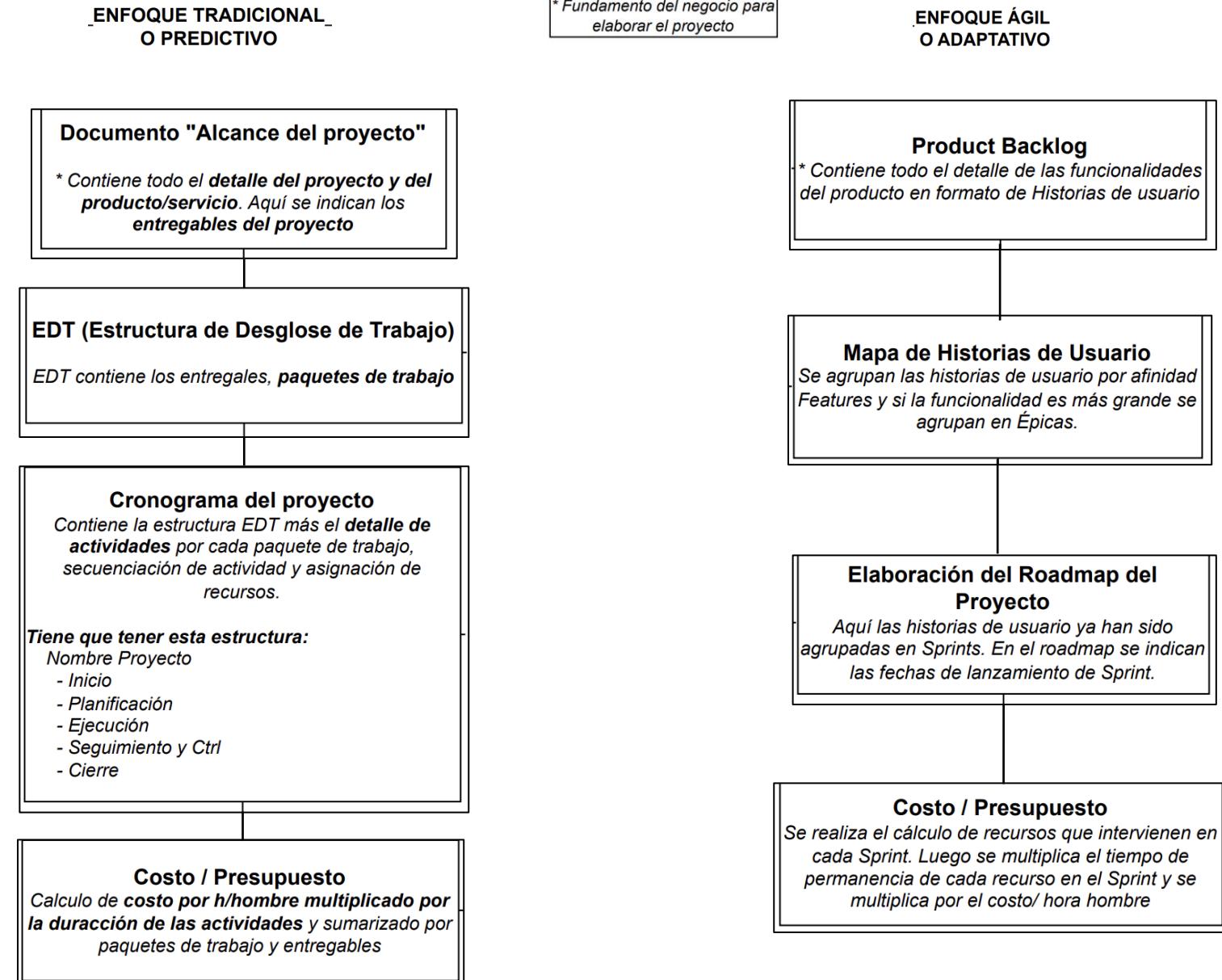
Fuente: Guía Práctica de Agilidad (PMI)

# Selección de ciclo de vida

Enfoque	Meta	Requisitos	Actividades	Entrega
<b>Predictivo</b>  Ejemplo: Casa Mi vivienda	Gestionar alcance, tiempo, costo	Fijo	Realizado una vez para todo el proyecto	Entrega única
<b>Iterativo</b>  Ejemplo: Rediseño de casa	Corrección de la solución	Dinámico	Repetidos hasta que esté correcto en teoría	Entrega única
<b>Incremental</b>  Ejemplo: Casa de playa por partes	Velocidad	Fijo	Realizados una vez para un incremento dado	Entregas frecuentes
<b>Adaptativo (Ágil)</b>  Ejemplo: Mejora de casa del albañil	Valor para el cliente	Dinámico	Repetidos hasta que esté correcto en la realidad	Entregas frecuentes

Adaptado: Guía Práctica de Agilidad (PMI)

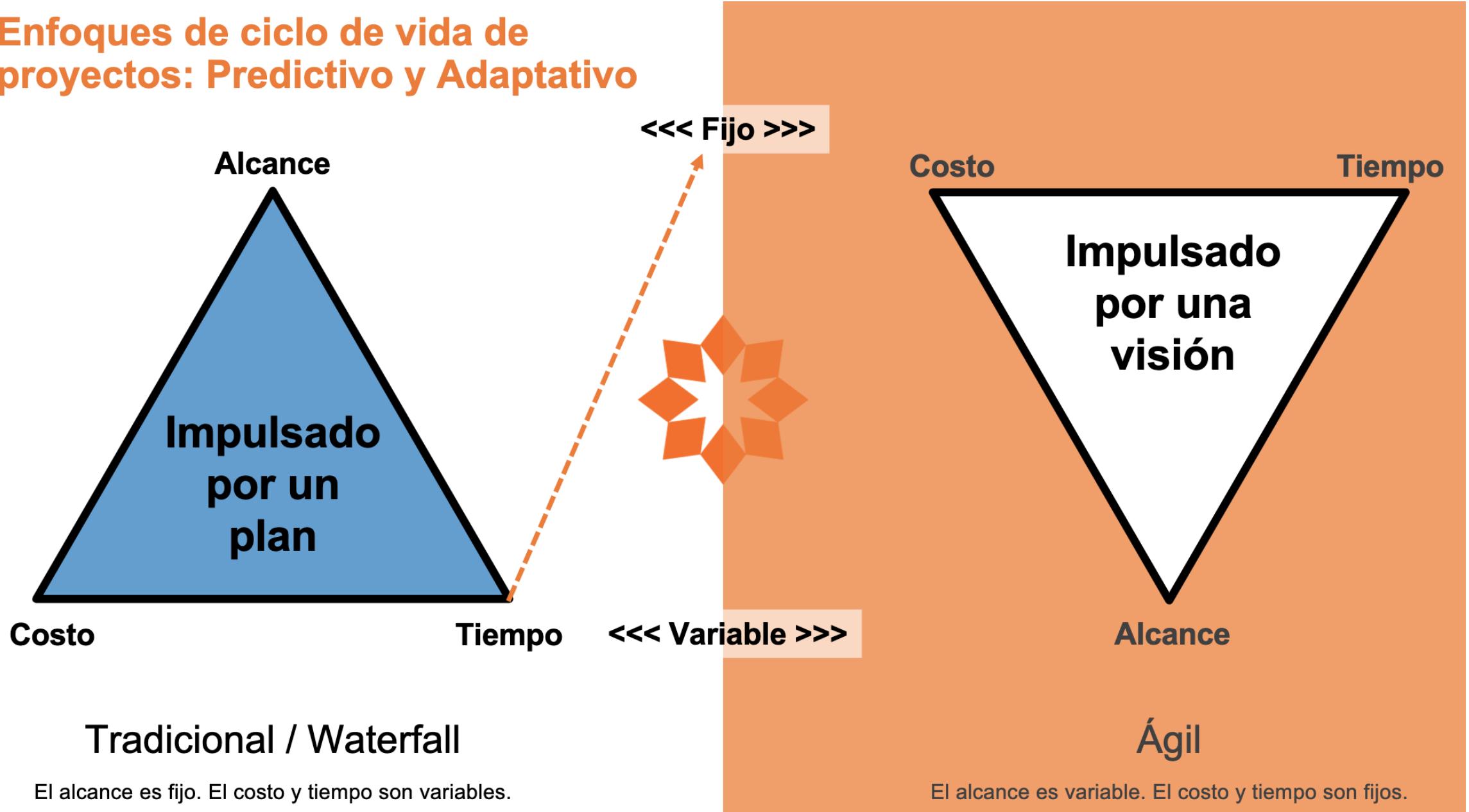
# Predictivo vs Adaptativo



# ¿Qué ciclo usar para afrontar estos proyectos?

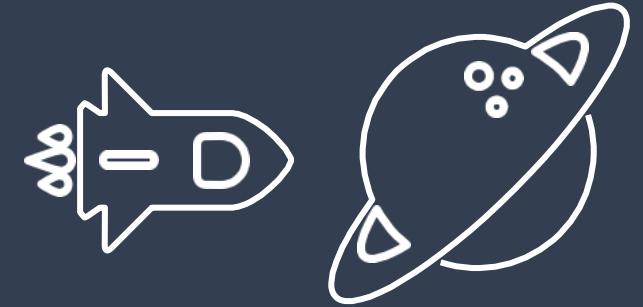


## Enfoques de ciclo de vida de proyectos: Predictivo y Adaptativo



# Metodologías Ágiles

*Qué son. Por qué surgen. El Origen.*



*Las metodologías ágiles son un conjunto de técnicas para gestionar y desarrollar proyectos en contraposición a las técnicas clásicas.*

# Metodologías Ágiles

A word cloud composed of various Agile methodology terms, each associated with a color and some additional text. The terms include:

- Priorities
- Estimation
- Product Owner
- User Stories
- Sprint
- Iterative Process
- Sprint Backlog
- Scrum Master
- Scrum
- Chickens
- Agile Manifesto
- Agile
- Pigs
- Business value
- Team Board
- Tasks List
- Development Team
- Events
- Product Backlog
- Retrospective
- Review Meeting
- Artifacts
- Knowledge Transfer
- Team
- Scrum Team
- Increment
- Iterations
- Planning Meeting
- Burndown Chart
- WAG
- Goal
- Release
- Production Ready Functionality
- Self Organizing
- Agile Methodology Requirements

# Problemas clásicos en el Desarrollo

- Cambios de contexto y de alcance
- Aparecen retrasos => No hay tiempo para pruebas
- Planificaciones poco realistas
- Cliente poco involucrado
- Falta de comunicación
- Equipo poco motivado
- No hay flexibilidad
- El resultado no es lo esperado por el cliente

**Resultado:** Equipo y cliente insatisfechos. Tiempo y dinero perdido.

## **Un poco de historia.**

Inicialmente llamadas “metodologías livianas” Luego, con el pasar de los años, en febrero de 2001, tras una reunión celebrada en Utah-EEUU, nace formalmente el término “ágil” aplicado al desarrollo de software.

En esta misma reunión participan un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software.

# Un poco de Historia

**1986**

En EEUU y Japón surge el concepto debido a la necesidad de salir al mercado muy rápido con requisitos muy novedosos.

**1993 - 1995**

Se documenta y formaliza el primer documento de Scrum para desarrollo ágil de software.

**2001**

Las personas más relevantes del desarrollo ágil escriben el Manifiesto Ágil donde se recogen sus 4 principios.

**... Antes de todo esto**

A finales del S. XIX ~ principios del S. XX surge el concepto **Lean Manufacturing** de la mano de Toyota.

# Manifiesto Ágil

- *Individuos e interacciones sobre procesos y herramientas*
- *Software funcionando sobre documentación excesiva*
- *Colaboración con el cliente sobre negociación contractual*
- *Respuesta ante el cambio sobre seguir un plan*

# Manifiesto Ágil.

- I. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.
- II. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
- III. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.
- IV. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.
- V. Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.
- VI. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
- VII. El software que funciona es la medida principal de progreso.
- VIII. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.
- IX. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
- X. La simplicidad es esencial.
- XI. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.
- XII. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

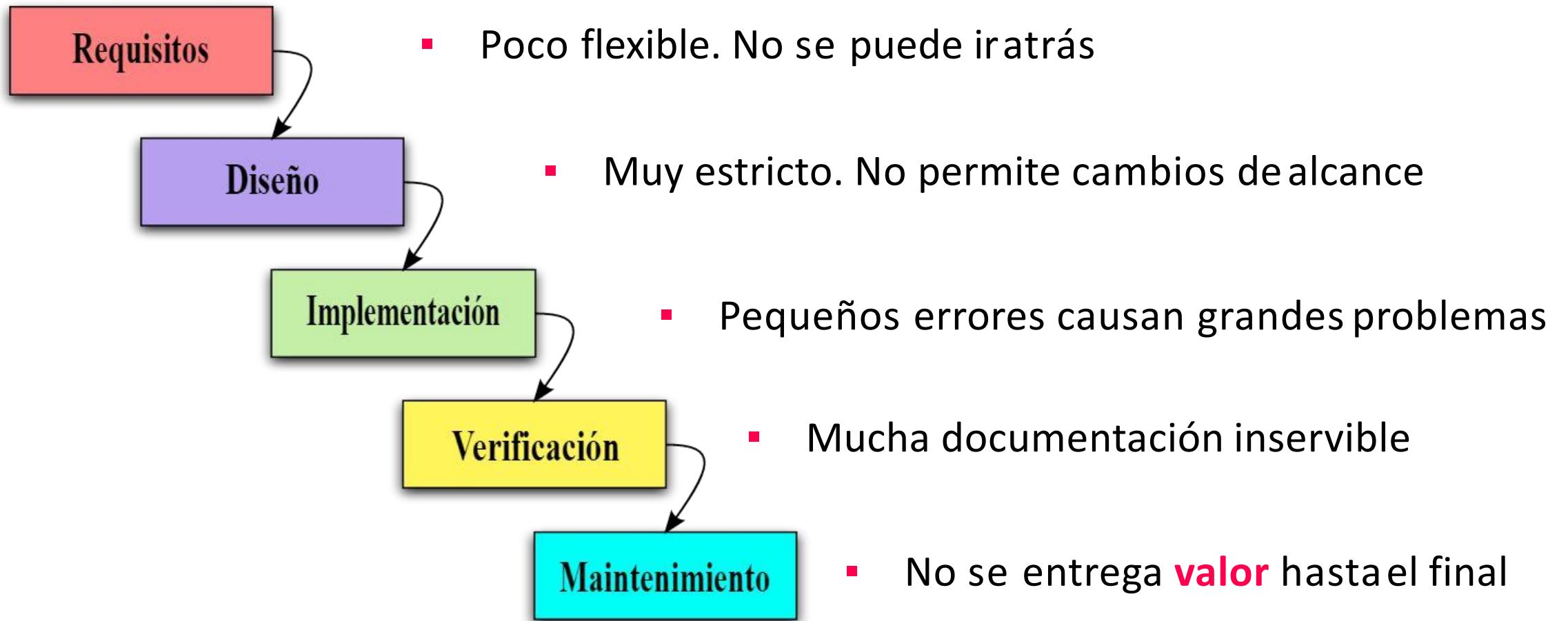
# ¿Por qué usar metodologías ágiles?

- Las metodologías ágiles de desarrollo están especialmente indicadas en proyectos con requisitos poco definidos o cambiantes.
- Estas metodologías se aplican bien en equipos pequeños que resuelven problemas concretos, lo que no está reñido con su aplicación en el desarrollo de grandes sistemas, ya que una correcta modularización de los mismos es fundamental para su exitosa implantación.
- Dividir el trabajo en módulos abordables minimiza los fallos y el coste.

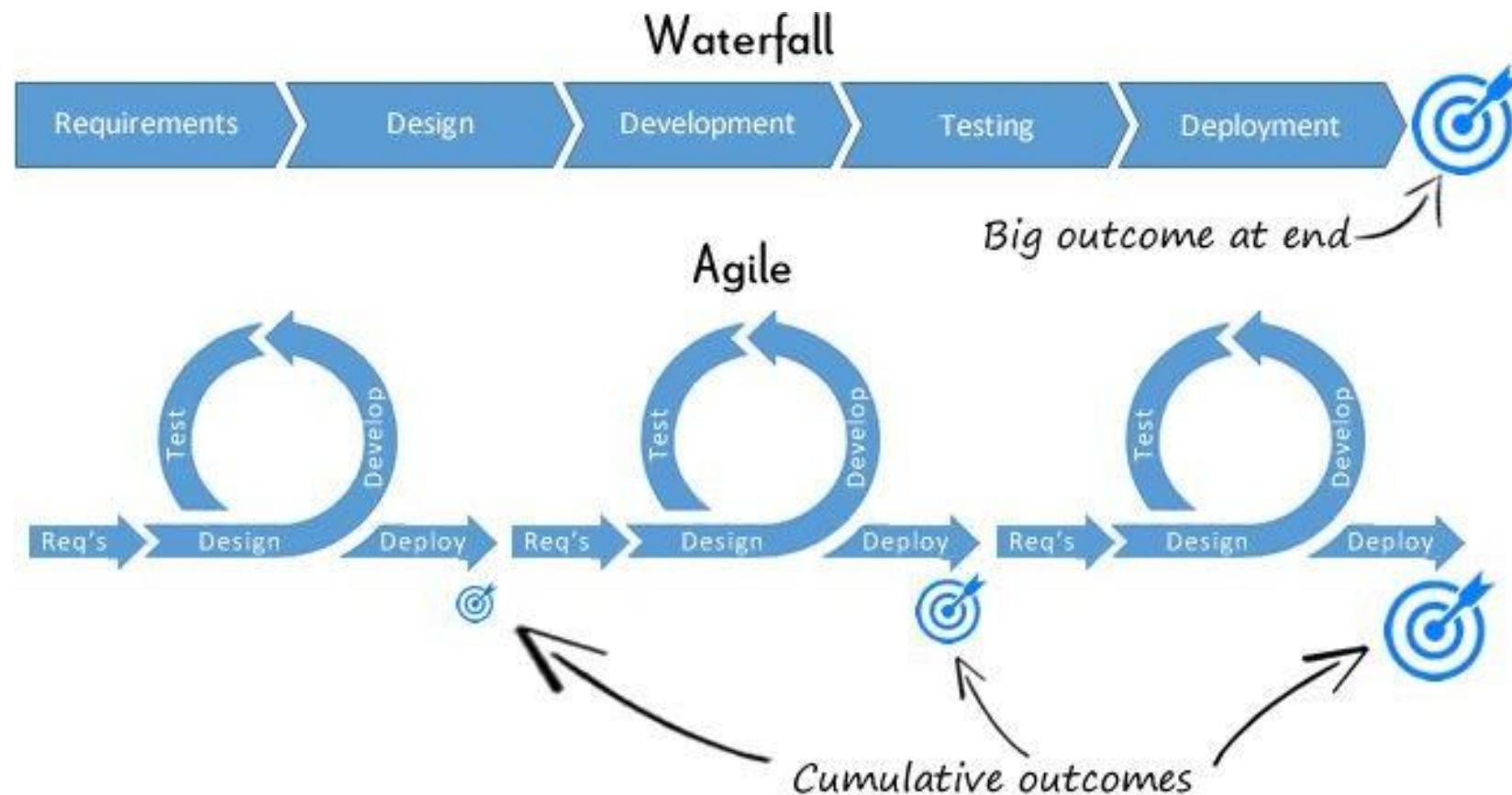
# Metodologías ágiles vs tradicionales

Qué aporta el agilismo, beneficios, cambios...

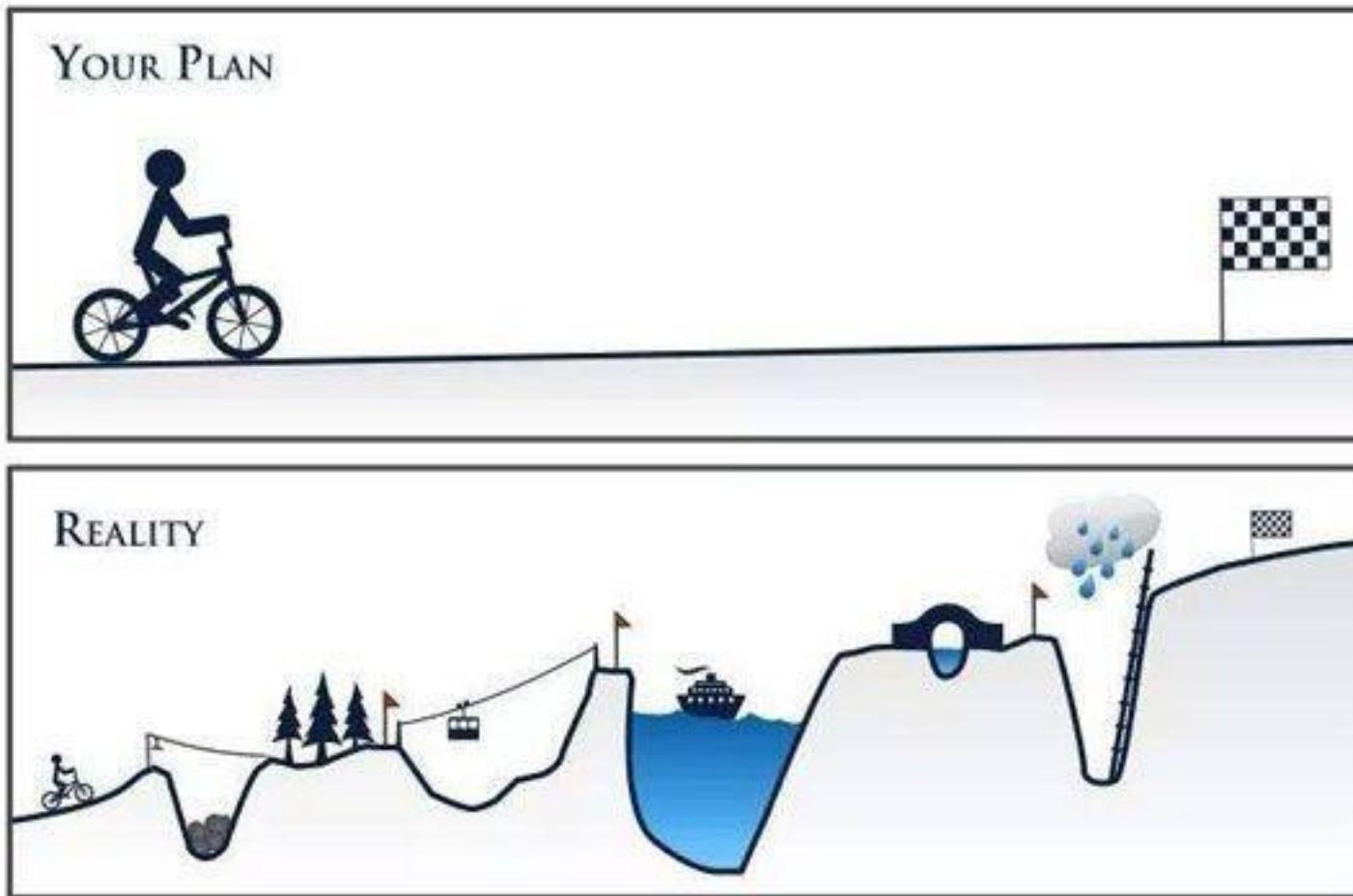
# Desarrollo en Cascada



# Cascada vs Agile

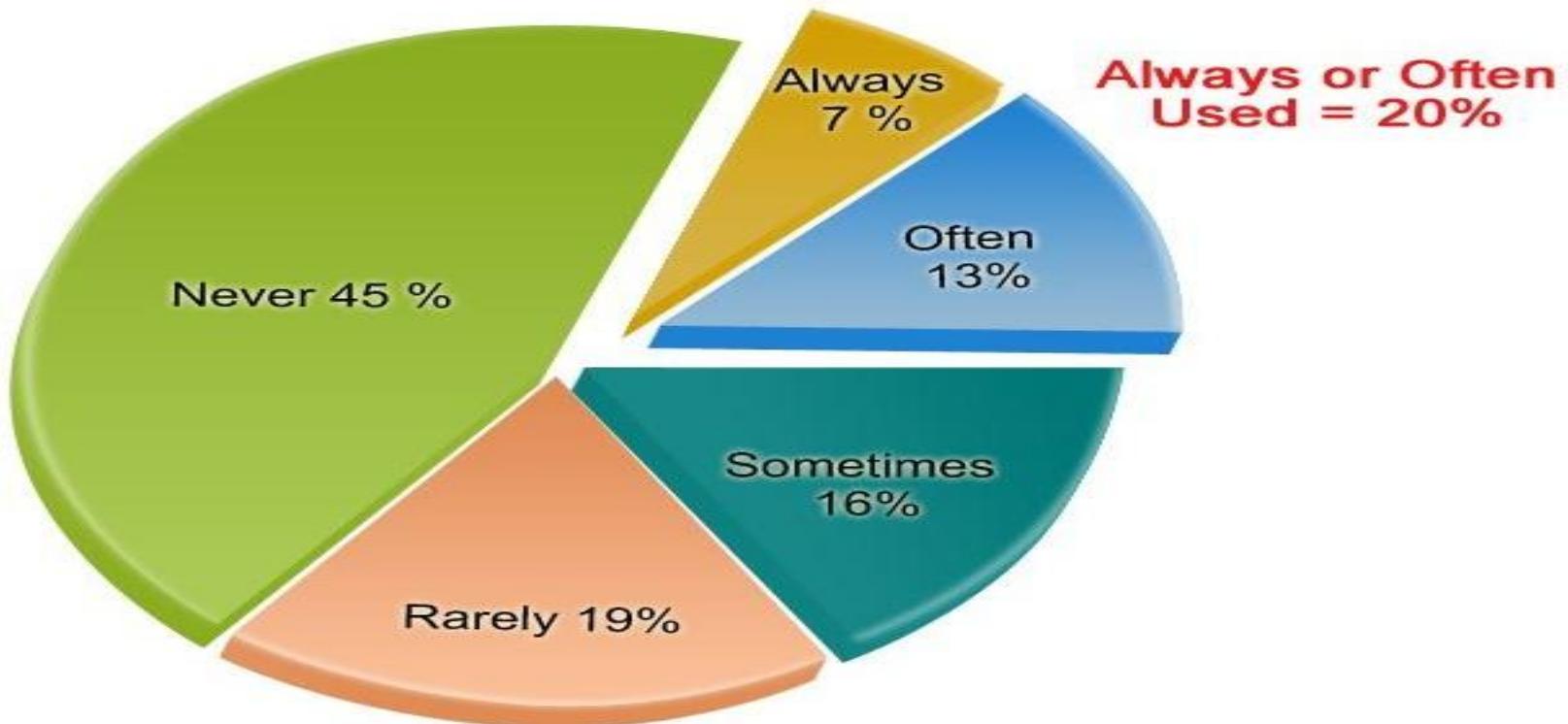


# Plan inicial vs Realidad



# Importancia del Feedback

Features and Functions Used in a Typical System



Standish Group Study - Reported at XP2002 by Jim Johnson, Chairman

Copyright © 2011 luuduong.com

# Diferencia entre Ágil y tradicional.

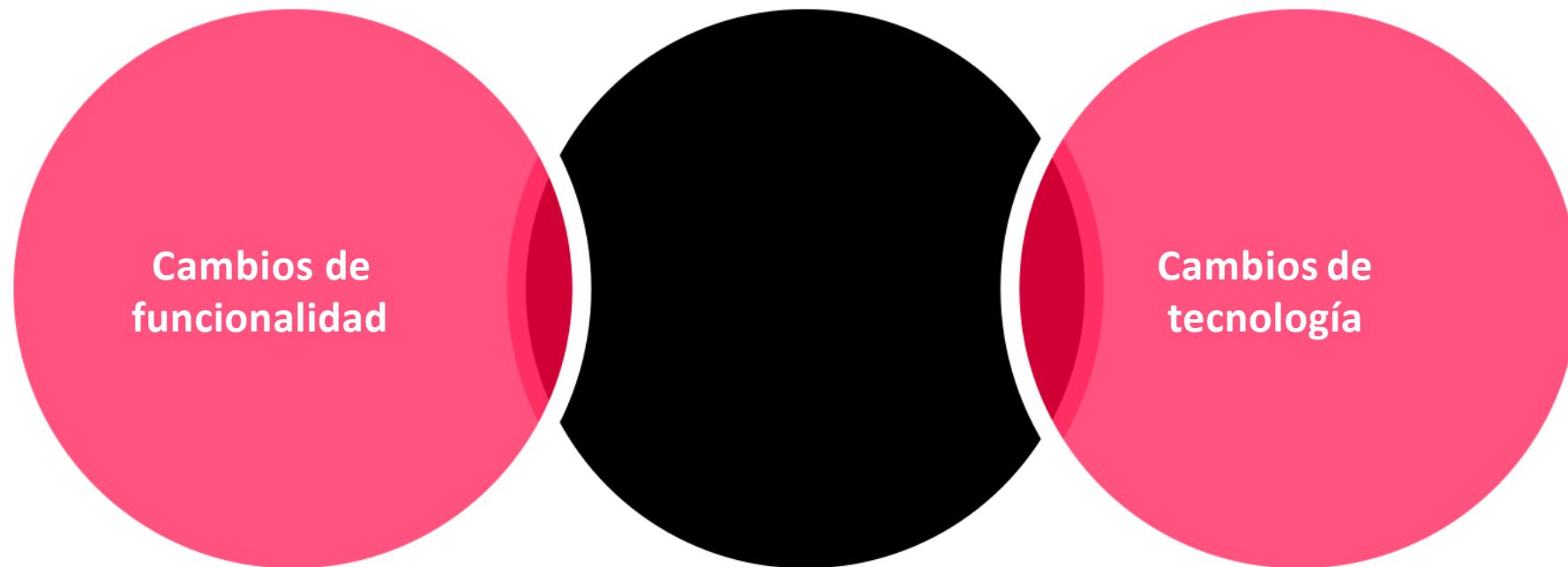
Metodologías ágiles	Metodologías tradicionales
Se basan en heurísticas provenientes de prácticas de producción de código	Se basan en normas provenientes de estándares seguidos por el entorno de desarrollo
Preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente por el equipo	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso muy controlado, numerosas normas
Contrato flexible e incluso inexistente	Contrato prefijado
El cliente es parte del desarrollo	Cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10)	Grupos grandes
Pocos artefactos	Más artefactos
Menor énfasis en la arquitectura del software	La arquitectura del software es esencial

*Se dedica mucho esfuerzo a  
alcanzar objetivos que aportan muy  
poco valor.*



*Dinero perdido + tiempo perdido = Cliente insatisfecho*

# **El gran enemigo Los cambios**



## Otros errores Típicos

- *No medir el avance o medirlo mal*
- *Añadir más personas creyendo que se irá más rápido*
- *No hacer pruebas desde el principio*
- *Creer que estamos construyendo una casa en vez de software*
- *No tener una visión global del estado actual*
- *Poca implicación del cliente*
- *Estimaciones sin técnicos*
- *Pérdida del foco*
- *No decir no*
- *No obtener feedback*
- *Herramientas inadecuadas para planificar*

¿Existe  
alguna  
alternativa?

Gestión  
Ágil



# Beneficios Metodologías Ágiles

## **Calidad**

Realizando pruebas desde el principio e iterando sobre el producto tras recibir el feedback.

## **Resultados**

Entregando algo tangible y que aporte valor desde la primera iteración.

## **Flexibilidad**

Permitiendo cambios de alcance, estimando y planificando de manera ágil.

## **Mantenibilidad**

Creando un software de calidad, con casos de prueba y una documentación asumible.

## **Eliminación de riesgos**

Validando cada entrega en sprints cortos y asegurando la calidad con casos de pruebas.

## **Motivación**

Trabajando de manera conjunta con el cliente, viendo crecer el producto final tras cada iteración.

# Definición del Producto



# Construcción Iterativa



# ¿Quién las usa?



# Metodologías Agiles

- XP (eXtreme programming)
- Scrum
- DSDM (Dynamic Systems Development Method)
- FDD (Feature Driven Development)
- Kanban

# Características Comunes

- Timeboxing
- Comunicación
- Patrocinador (Sponsor)
- Equipo de Alto rendimiento
- Continuous Integration

# **Casos de uso vs Historias de usuario**

## **Casos de uso**

Descripción de todos los pasos que se deben llevar a cabo para realizar una acción.

Especificación de interacciones entre los actores y el sistema.

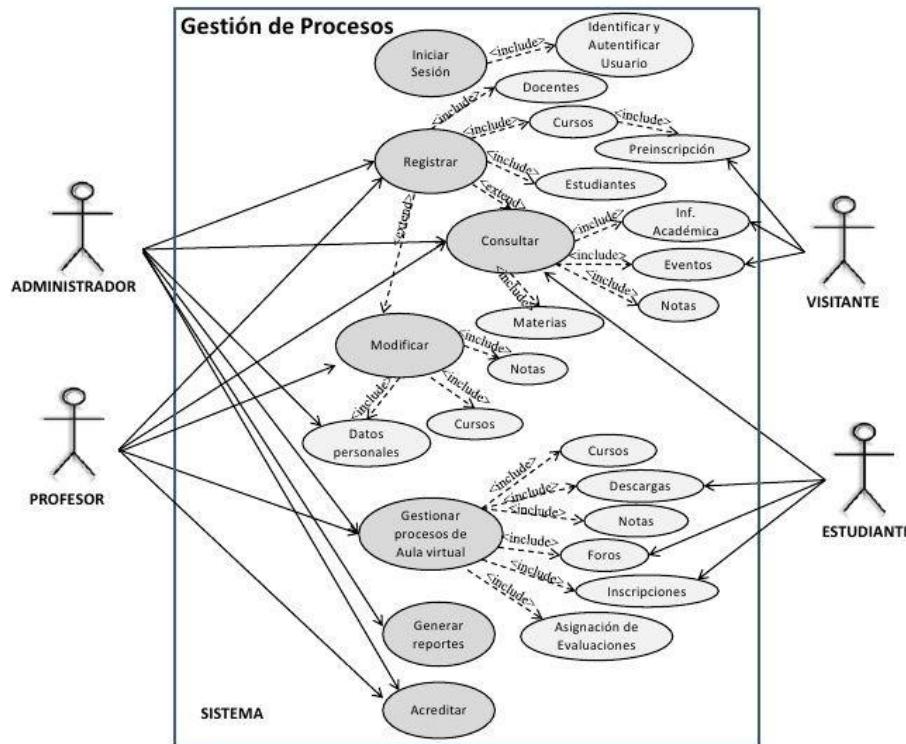
## **Historias de usuario**

Definición corta de una funcionalidad, que debe poder escribirse en una nota adhesiva.

Lenguaje sencillo de entender por el equipo y el cliente.

# Casos de uso vs Historias de usuario

## Casos de uso



## Historias de usuario

Historia: Agregar comentarios

Como: Lector del Blog

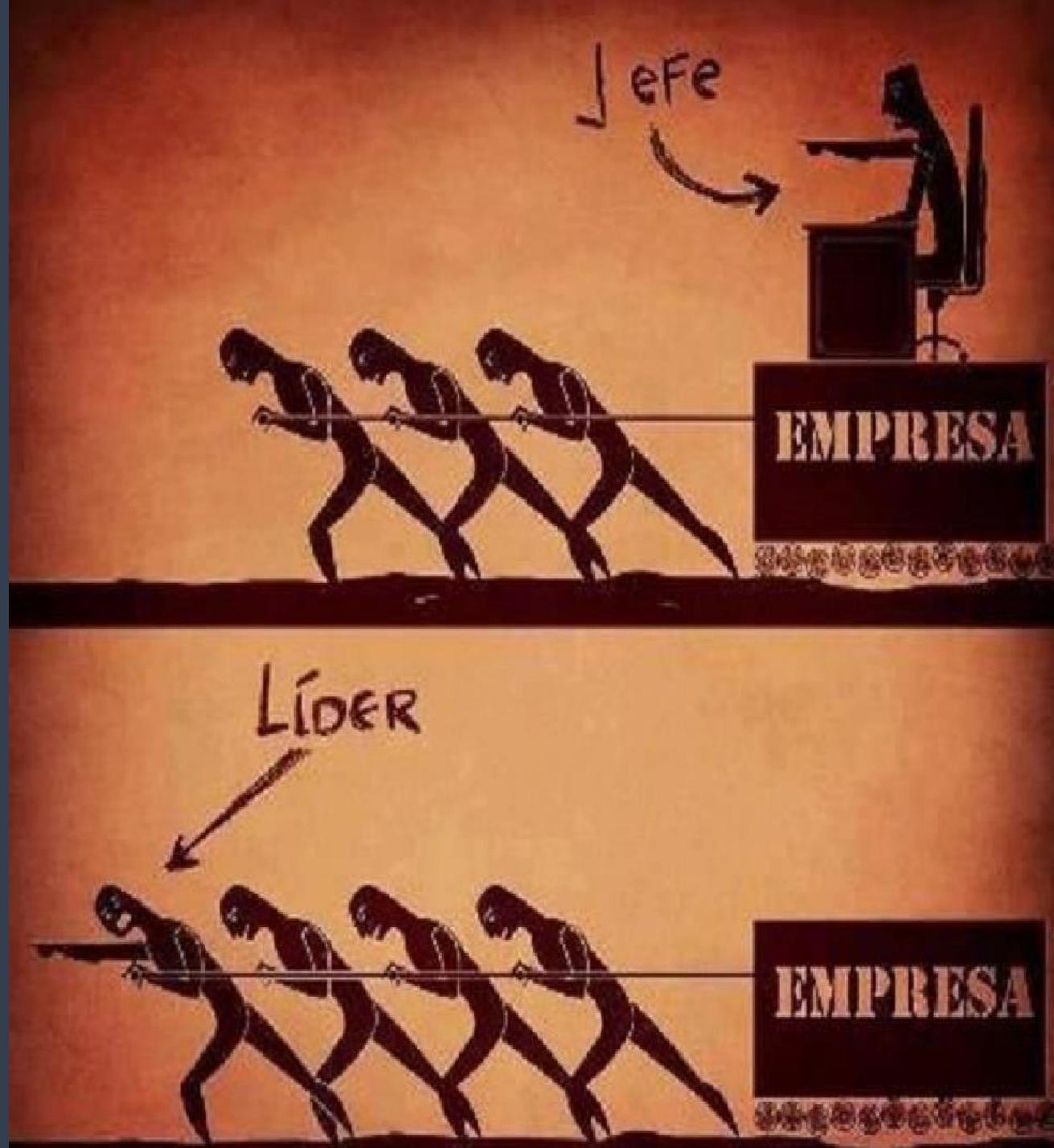
Quiero: adicionar comentarios a las entradas y recibir alertas cuando otros hagan comentarios

Para: mantenerme en contacto con los demás usuarios del blog

3

# Jefe vs Líder

- Desaparece el jefe autoritario por el líder con conocimientos que guía al equipo.
- Soluciones vs problemas
- Confianza vs miedo
- Convencer vs imponer



# Scrum

¿Qué es? Roles, prácticas...

¿Qué es?

SCRUM



# Qué es Scrum

- Marco de trabajo para desarrollos ágiles
- Desarrollo incremental vs planificación y ejecución completa
- Equipos auto organizados
- Paralelización de las fases de desarrollo vs fases secuenciales
- Priorización de los requisitos que más valor aporten
- Mejora continua



Scrum.org

# La importancia de Priorizar

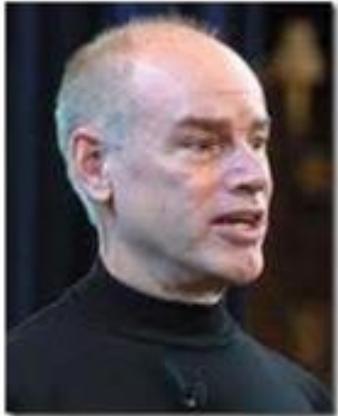
- Es una responsabilidad del Product Owner
- Se debe priorizar por el valor que aporta cada historia
- No se debe priorizar por la complejidad para desarrollarlas
- Existen muchas técnicas, como por ejemplo:
  - Modelo Kano:
    - ▶ Requisitos obligatorios (Básicos)
    - ▶ Requisitos deseados (Esperados)
    - ▶ Requisitos no esperados (Inesperados)
    - ▶ Indiferentes (No aportan valor)
  - MoSCoW: (Must, Should, Could y Won't)

# La necesidad de Estimar

- Es una responsabilidad de todo el equipo
- Todas las tareas deben ser estimadas
- Estimación basada en el conocimiento y en la experiencia
- Estimar en puntos y conocer la velocidad del equipo
- Planning Poker:
  - Se utiliza la secuencia de Fibonacci
  - Se explica la historia y se resuelven dudas
  - Se busca unanimidad y consenso



# Scrum...



- ...es un **framework** ágil y simple para el desarrollo de productos complejos en ambientes complejos;
- ...no es un proceso o técnica: dentro de Scrum se pueden emplear **varios procesos y técnicas**;

- ...utiliza el abordaje **iterativo e incremental** para mejorar la previsibilidad y la gestión de riesgos;
- ...vuelve los problemas de las prácticas de desarrollo **transparentes**, para que se puedan mejorar;



# Scrum...



- ...focaliza el **orden** del trabajo basado en el mayor **valor de negocio** para el cliente
- ...genera entregas de valor al cliente, frecuentemente y de forma temprana;

- ...utiliza equipos **auto-organizados**, que definen las mejores formas de desarrollar las funcionalidades de mayor valor
- ...es orientado a **valor** y no a planes
- ...utiliza inspección y adaptación para la **mejora continua** del producto y de los procesos de desarrollo

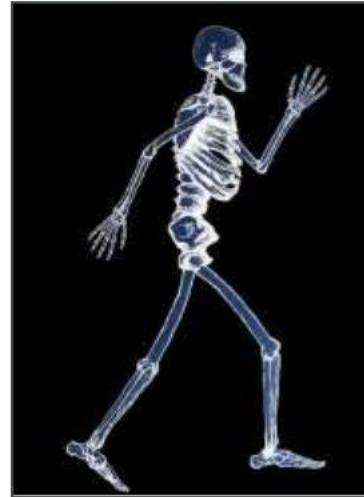


# Pilares de Scrum

- #1: Transparencia: ver y entender



- #2 Inspección: investigar
- #3 Adaptación: mejorar



# Equipo de Scrum

- **Product Owner**

- Garantiza y maximiza el ROI del cliente a partir del trabajo del Equipo

- **Equipo de Desarrollo**

- Genera valor para el cliente construyendo incrementos del producto con alta calidad

- **ScrumMaster**

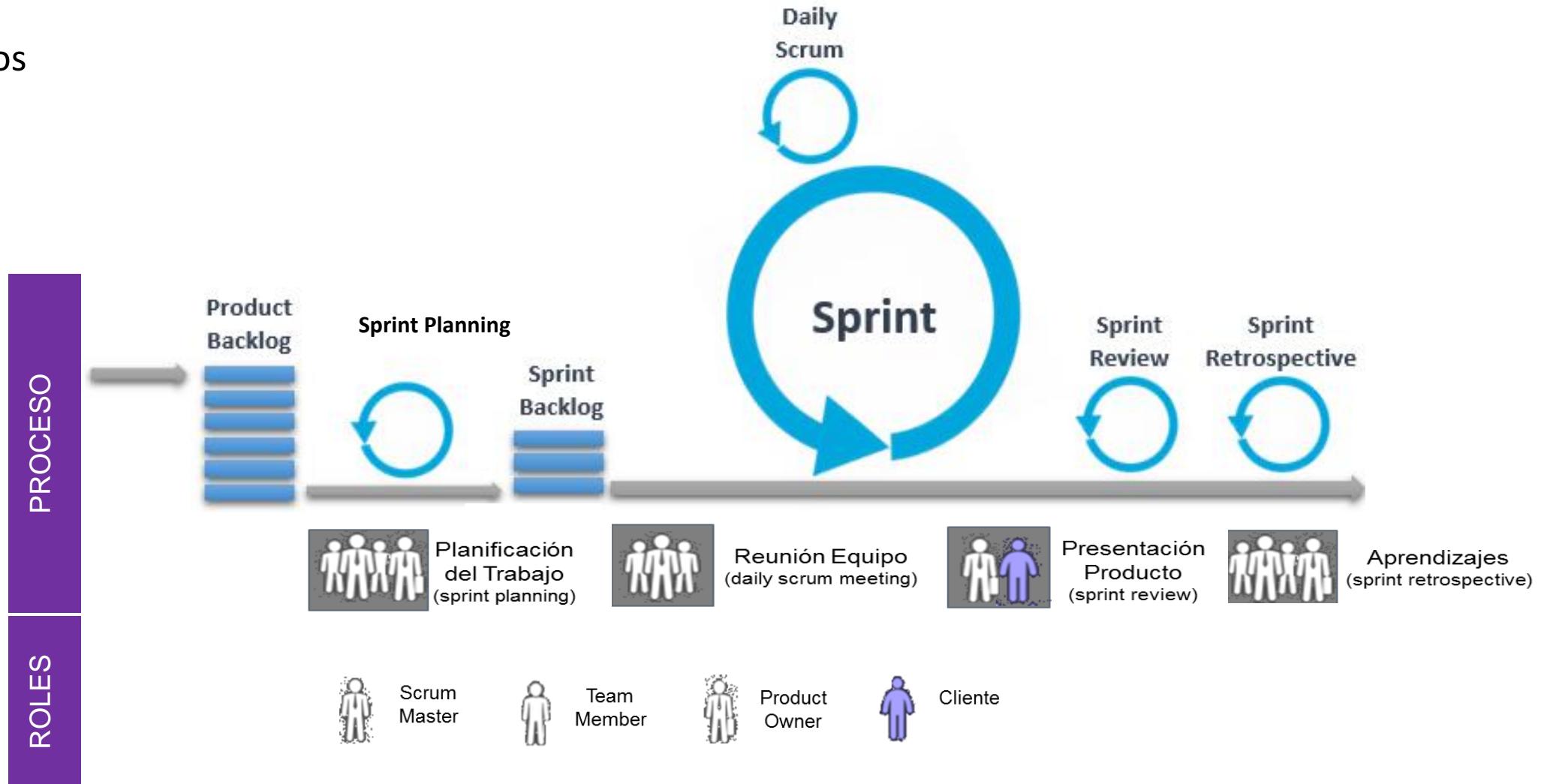
- Garantiza que los valores prácticas y reglas de Scrum están siendo comprendidos y seguidos

¡El nombre **Scrum** viene del Rugby!

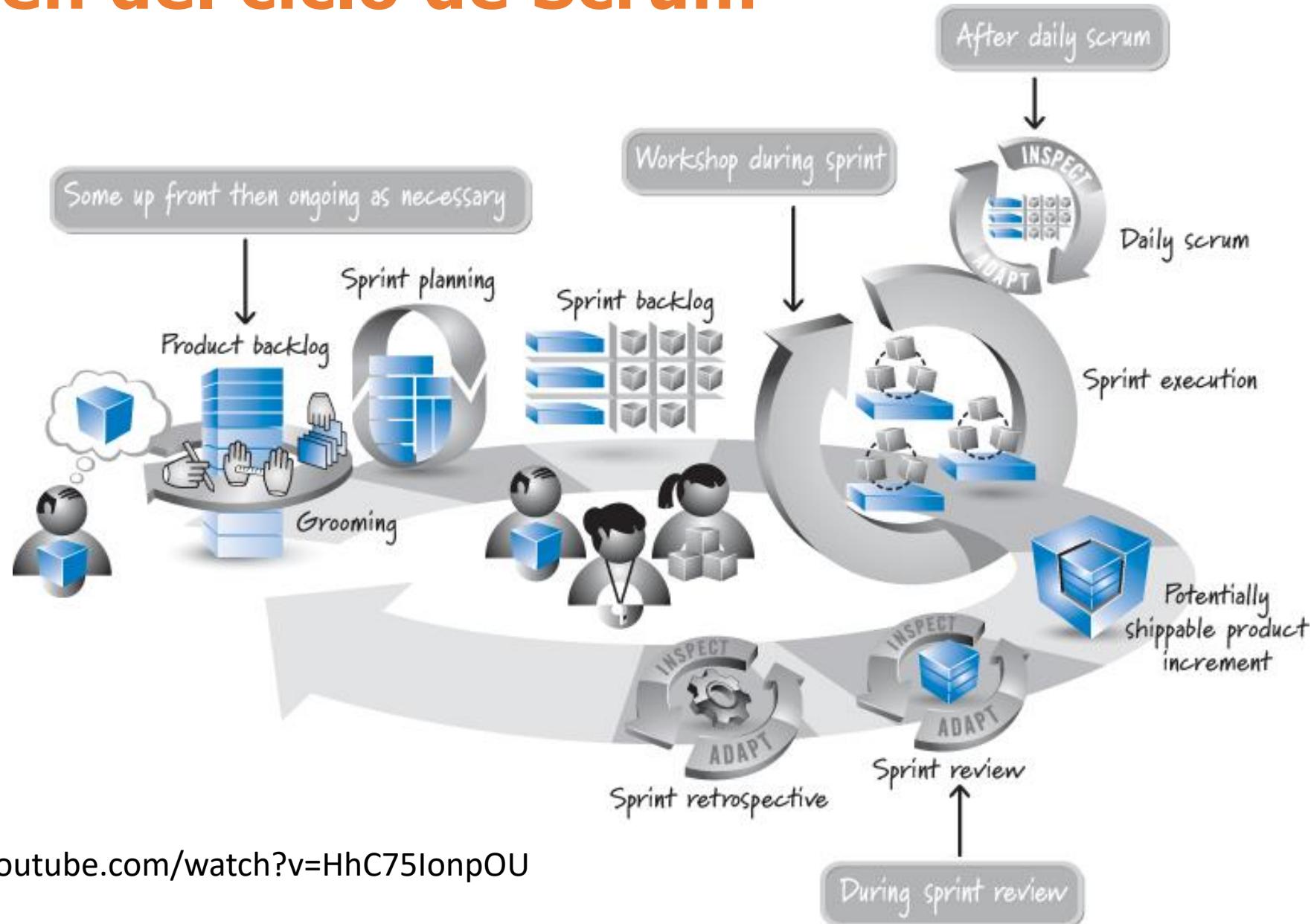


# Scrum

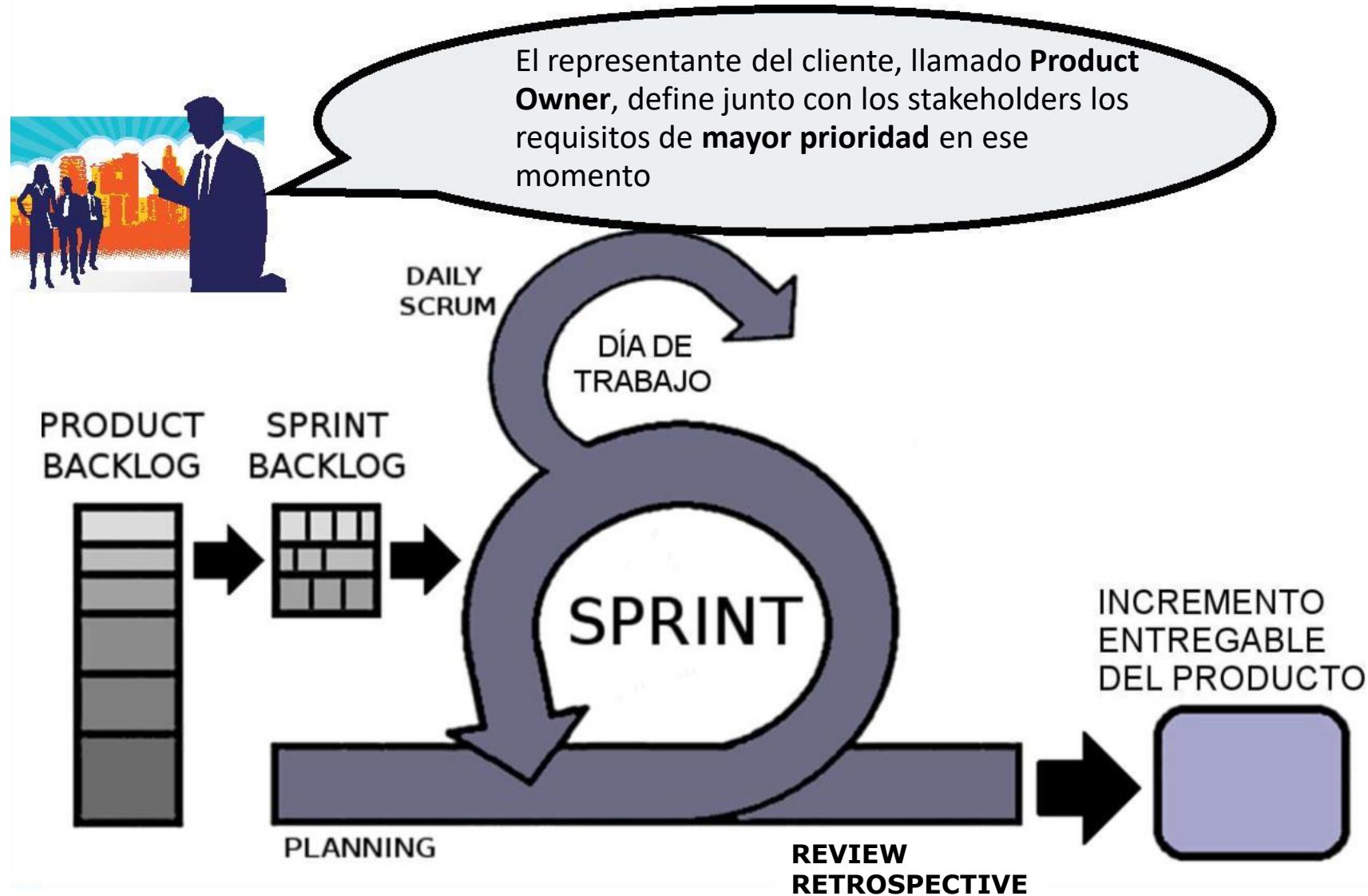
## Procesos



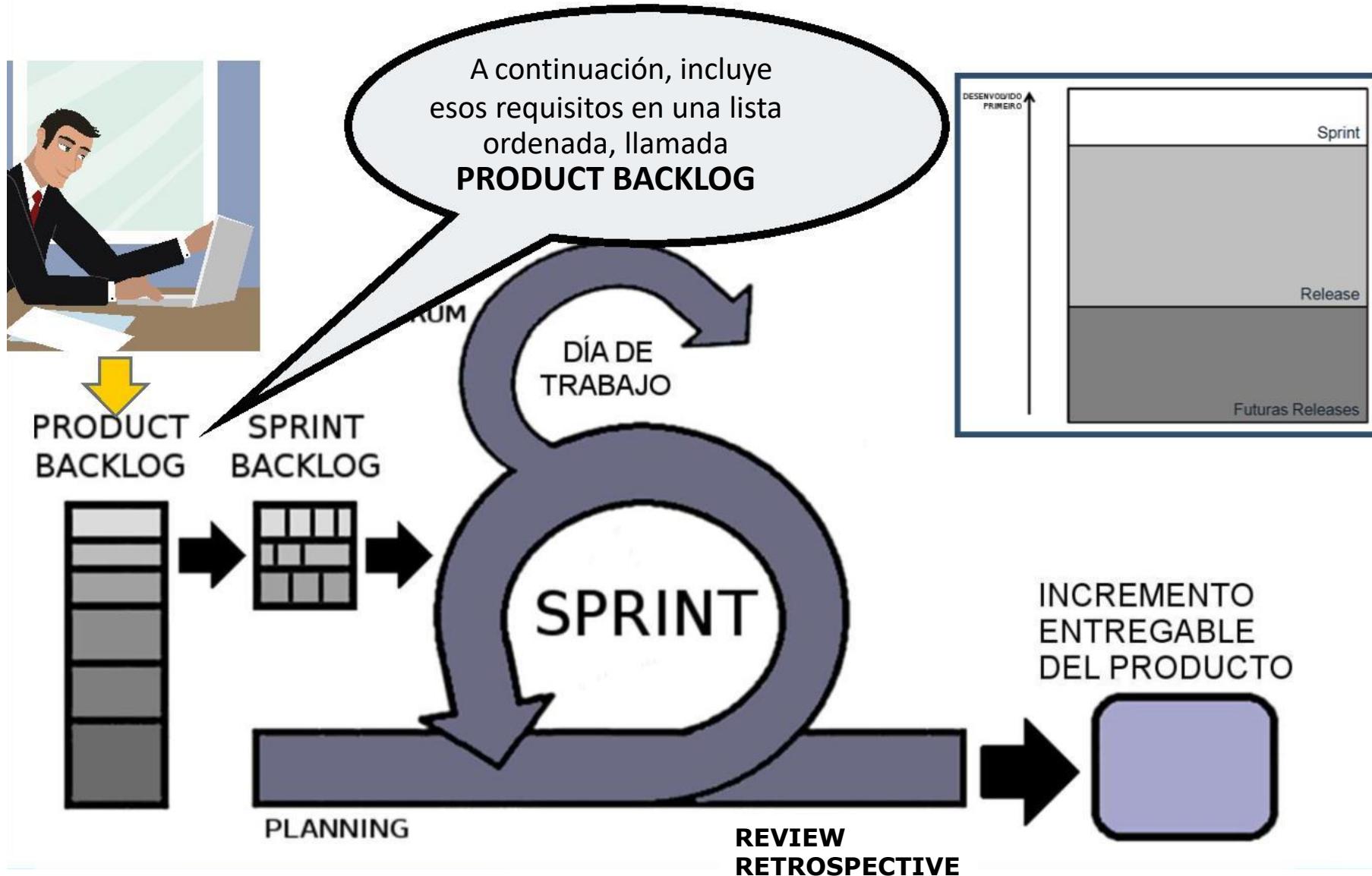
# Resumen del ciclo de Scrum



# Scrum: Product backlog



# Scrum: Product backlog



# Scrum: Sprint

- El Sprint es la **iteración (ciclo)** de desarrollo
  - Sprint Planning Meeting
  - Trabajo de Desarrollo
  - Sprint Review
  - Sprint Retrospective
- Cada Sprint debe contar con una **meta de negocios**
- Tienen duración fija (de 1-2 semanas a 1 mes) y ocurren uno atrás del otro
- No debe haber **ningún cambio** que afecte el Objetivo del Sprint

# Scrum: Sprint

- Cada Sprint debe tener como resultado un incremento **entregable** del producto que satisfaga el **objetivo del Sprint**
- Al final del Sprint, un trabajo **entregable** debe estar **terminado**
- El deadline **no puede ser cambiado**. Sólo puede variar su alcance (siempre que no afecte el objetivo del Sprint)
- Durante el Sprint, el P. O. debe estar disponible para el Equipo de Desarrollo

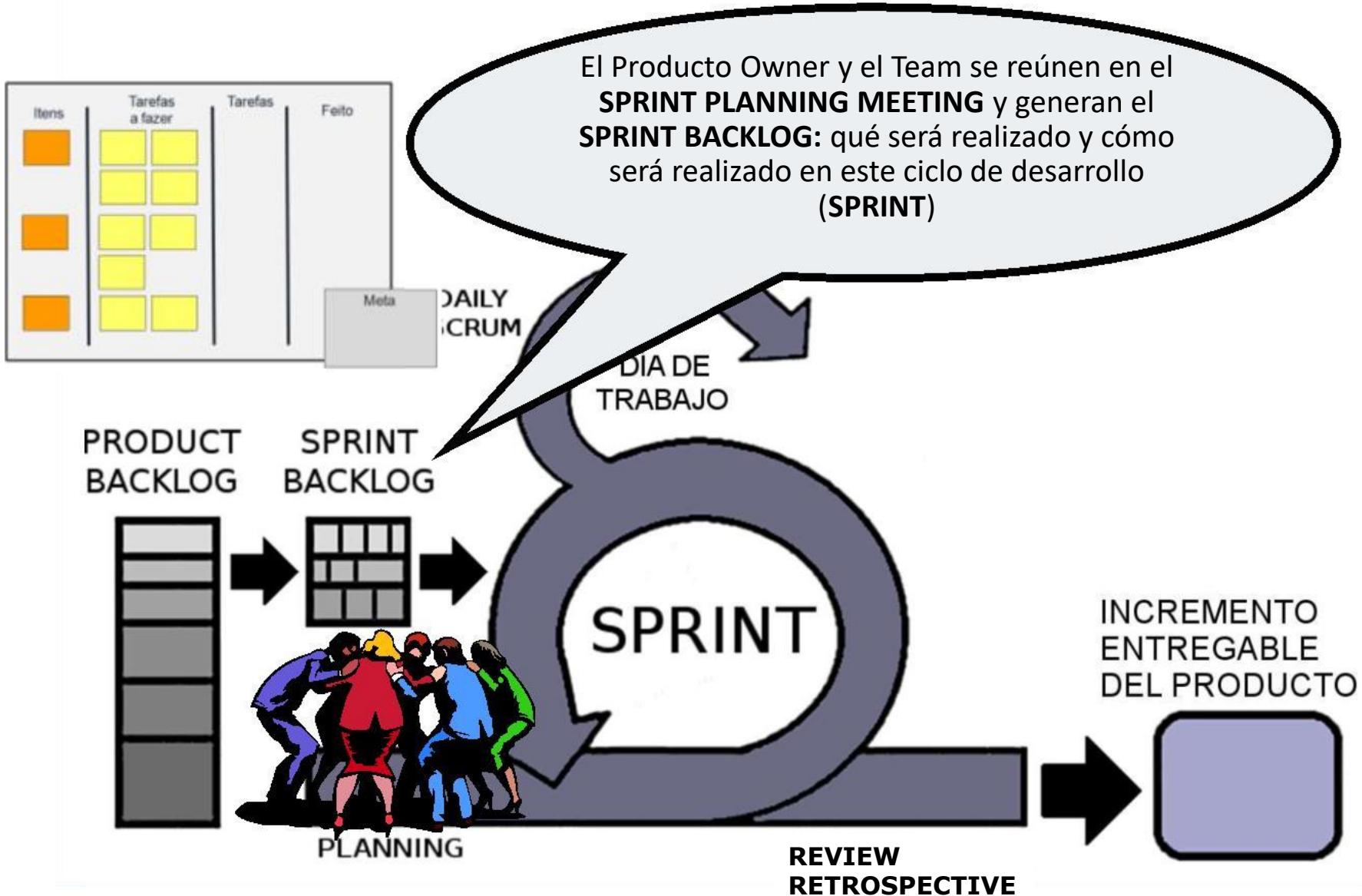
# Scrum: Sprint

- El Sprint puede ser **cancelado** si la Meta del Sprint pierde el sentido
- **Sólo el Product Owner** puede decidir sobre la cancelación del Sprint
- Pero es una **excepción**, debe ocurrir raramente
- Los ítems listos (“done”) son revisados y pueden ser aceptados. Inmediatamente, se inicia un nuevo Sprint

# Scrum: Sprint

- ¡El tamaño del Sprint es fijo! (1-4 semanas) Sólo puede ser modificado si es detectada la necesidad en el Sprint Retrospective
- Horizonte suficientemente corto para que los **cambios necesarios no modifiquen la meta** del Sprint
- **Sprints cortos:** cambios muy frecuentes, entregas más frecuentes, proyectos cortos
- **Sprints largos:** cambios menos frecuentes, overhead de reuniones

# Scrum: Sprint planning

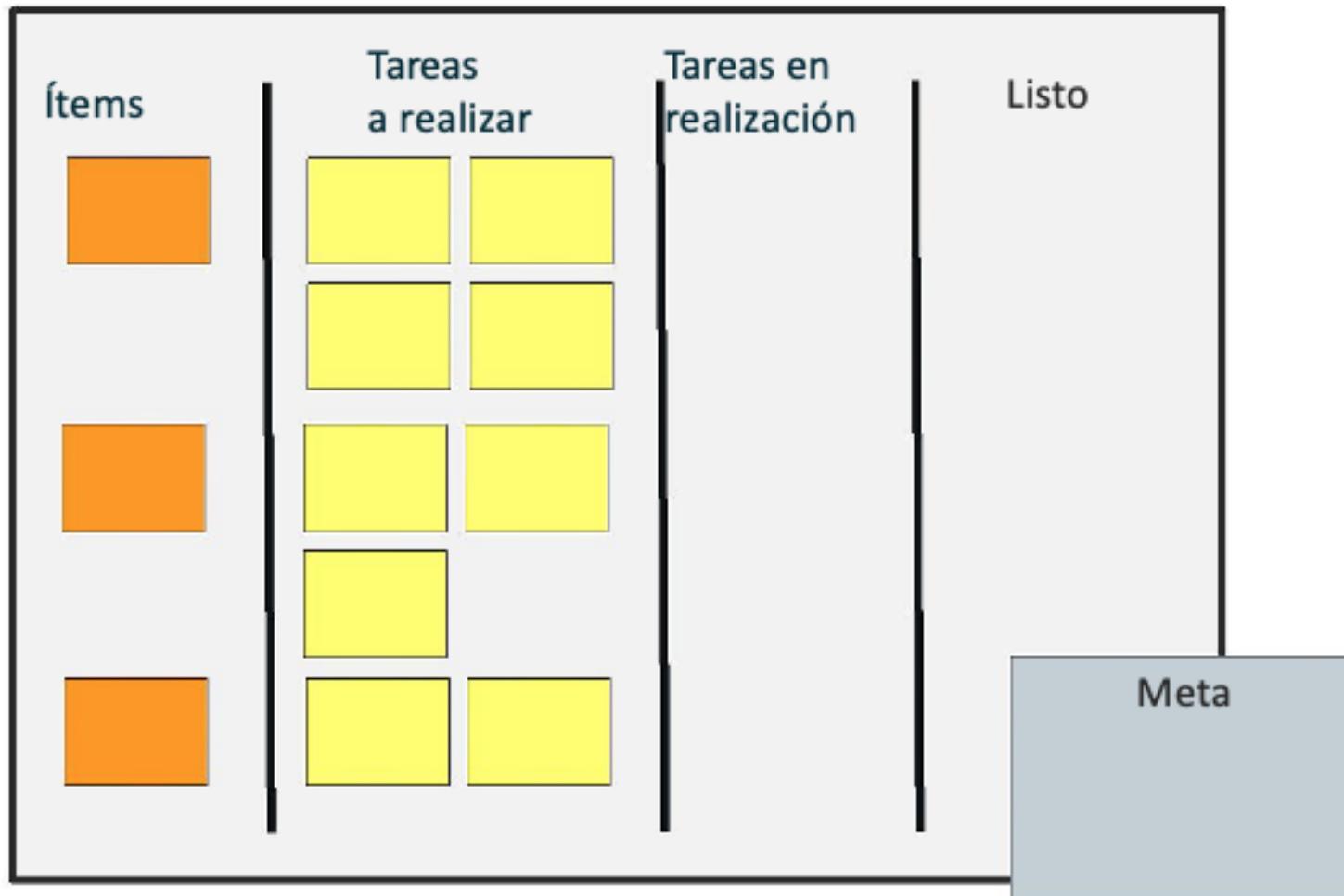


# Scrum: Sprint planning

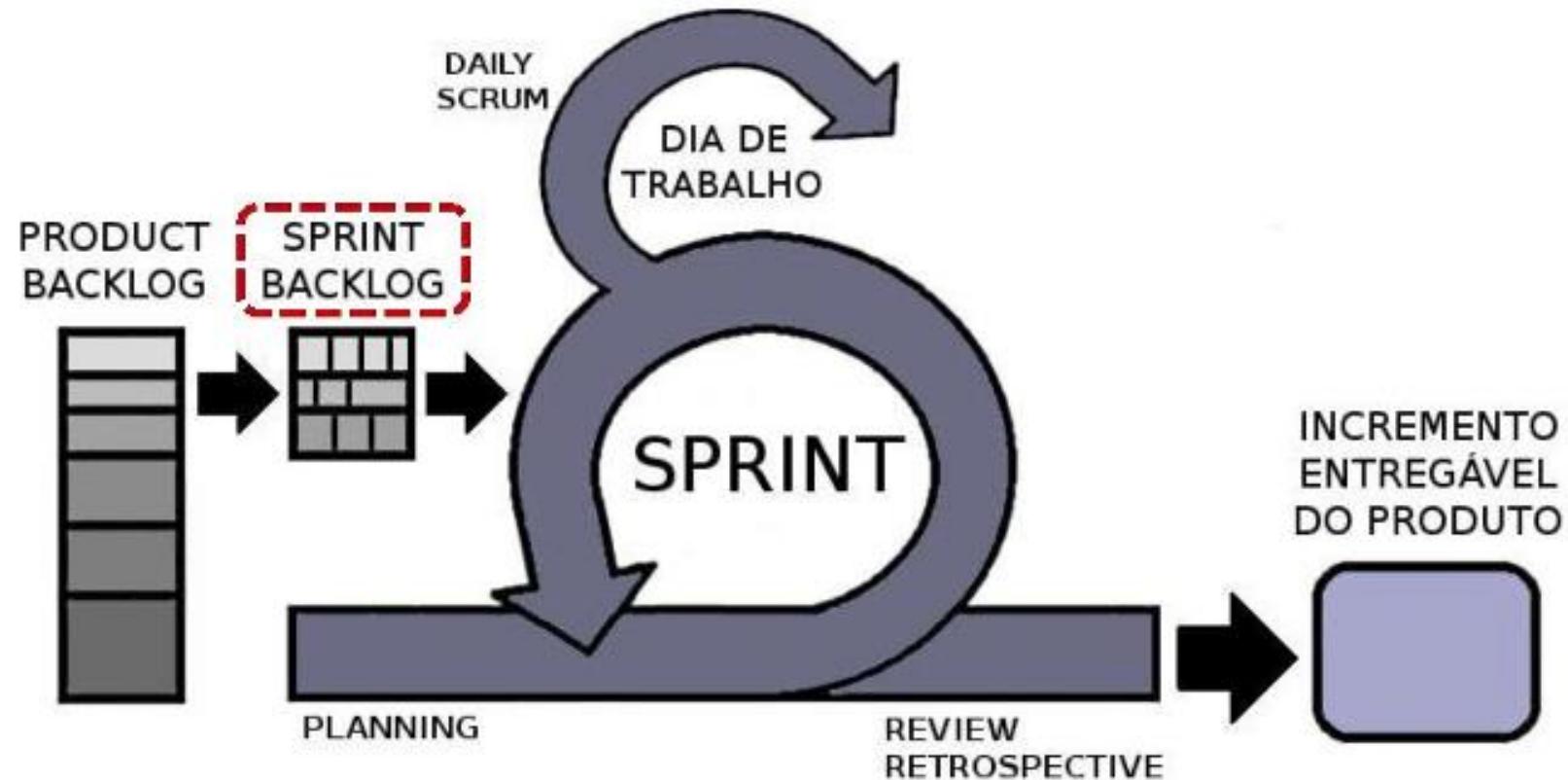
- Planificación de la iteración:
  - El Equipo de Desarrollo y el Product Owner definen los ítems del Product Backlog que serán implementados en el Sprint y los dividen en tareas – **Sprint Backlog**
  - Reunión de 8 h (Sprints de 1 mes) o 5% del Sprint
    - 1a. Parte: **¿Qué?**
      - Selección de los ítems más prioritarios del Product Backlog que serán implementados
      - Definición de el **Objetivo del Sprint**
      - El Product Owner debe estar presente
    - 2a. Parte: **¿Como?**
      - El Equipo de Desarrollo divide los ítems en tareas y estima el tiempo (cuando se utiliza) para la realización de cada tarea

# Scrum: Sprint planning meeting

- Resultado: Sprint Backlog inicial + Objetivo



# Scrum: Sprint backlog



# Scrum: Sprint backlog

- Está formado por una **lista de los ítems que serán desarrollados** durante el Sprint, las tareas correspondientes, su evolución y las estimaciones
- Los ítems son **seleccionados del Product Backlog** en el Sprint Planning Meeting
- Cada ítem es dividido en tareas. Parte de las tareas es definida en el Planning y parte a lo largo del Sprint

# Scrum: Sprint backlog

- Las tareas pueden ser estimadas o no, pero debe ser trazado el Sprint Burndown
- El Sprint Backlog es **modificado** a lo largo del Sprint
  - las estimaciones (cuando las hay) son actualizadas
  - las tareas pueden surgir para los ítems ya existentes
- Debe haber **alta visibilidad**
- Pertenece al Equipo de Desarrollo

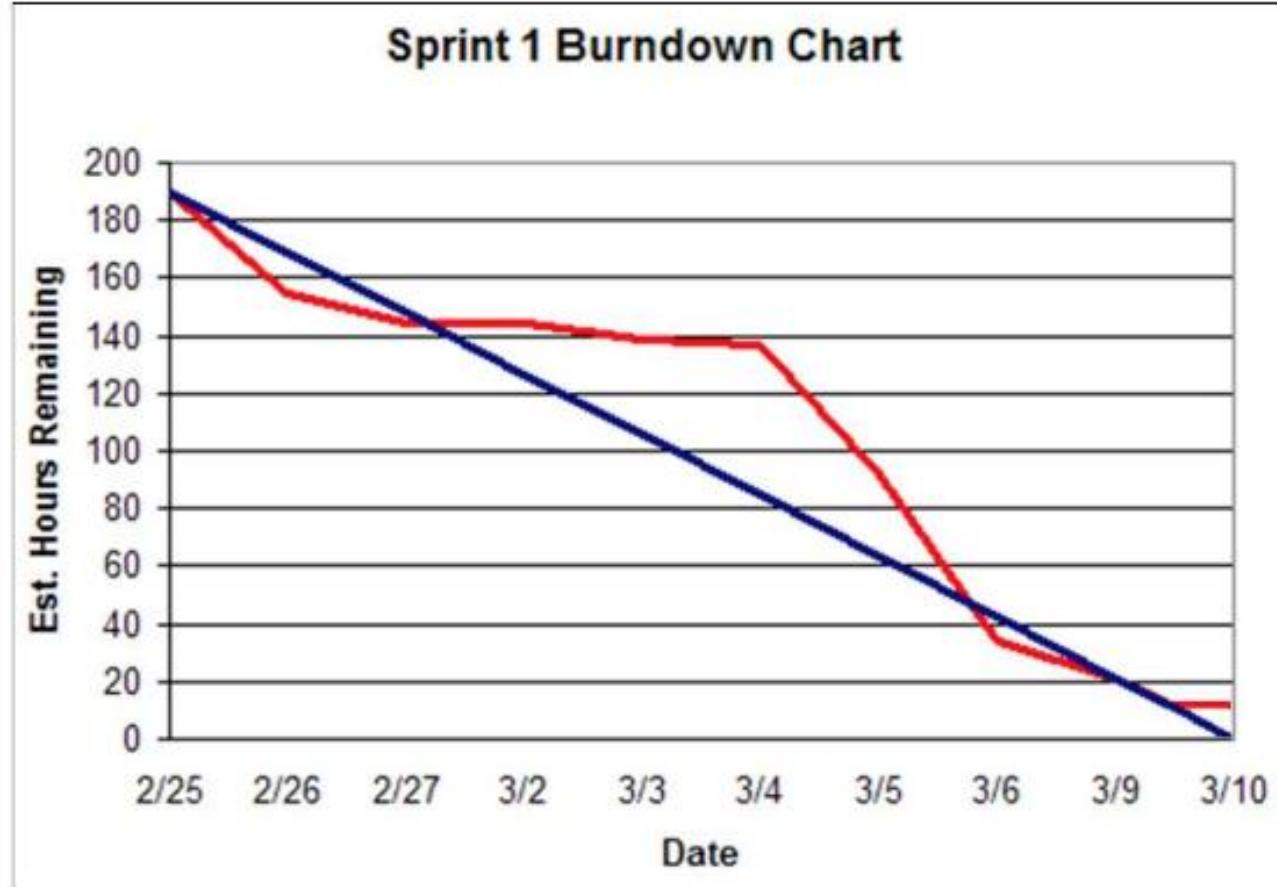
# Scrum: Sprint backlog - tareas

- En la segunda parte del Sprint Planning, los miembros del Equipo de Desarrollo **estiman las tareas del Sprint Backlog**
- **Estimación por horas:** número de horas previstas para desarrollar la tarea
- **Estimación T-Shirt Sizing:** P, M, G, GG
- Algunos Equipos de Desarrollo **no estiman** sus tareas
- De preferencia, cada tarea es < 1 día y > 2 horas
- Las estimaciones deben ser actualizadas siempre que sea pertinente

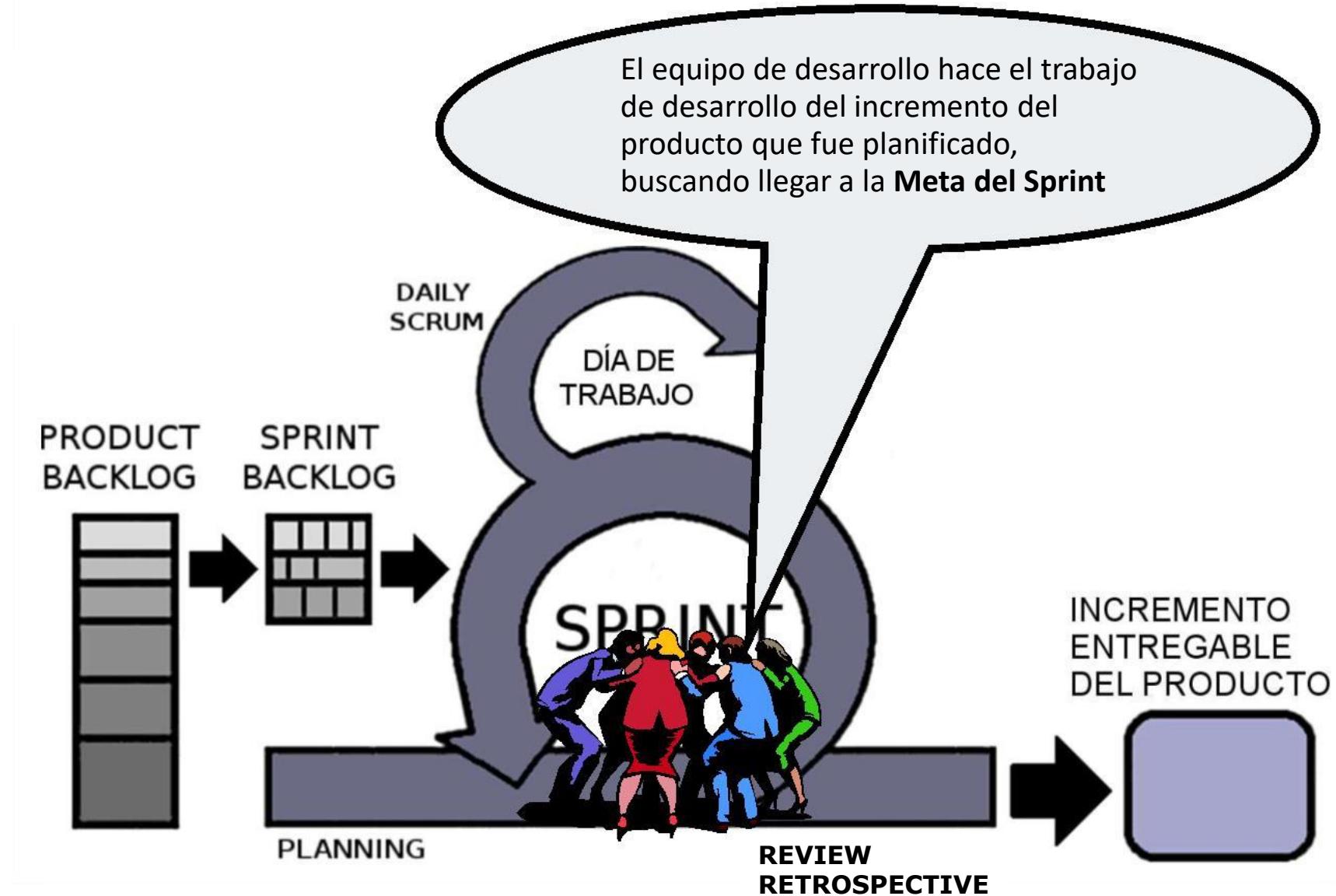
# Scrum: Sprint burndown

- El **Sprint Burndown** es un gráfico que muestra el **trabajo restante estimado para las tareas** del Sprint Backlog en el **tiempo**
  - Y: trabajo restante estimado para las tareas
    - suma de las horas estimadas restantes de las tareas
    - (Ó) suma de los valores restantes correspondientes a P, M, G, GG (por ejemplo: 2, 4 ,8, 16)
    - (Ó) número de tareas restantes
  - X: tiempo
    - Días del Sprint
- Sirve para acompañar el **progreso** de un **sprint**
- Inicialmente, es realizado en el Sprint Planning Meeting y debe ser actualizado a cada día del Sprint

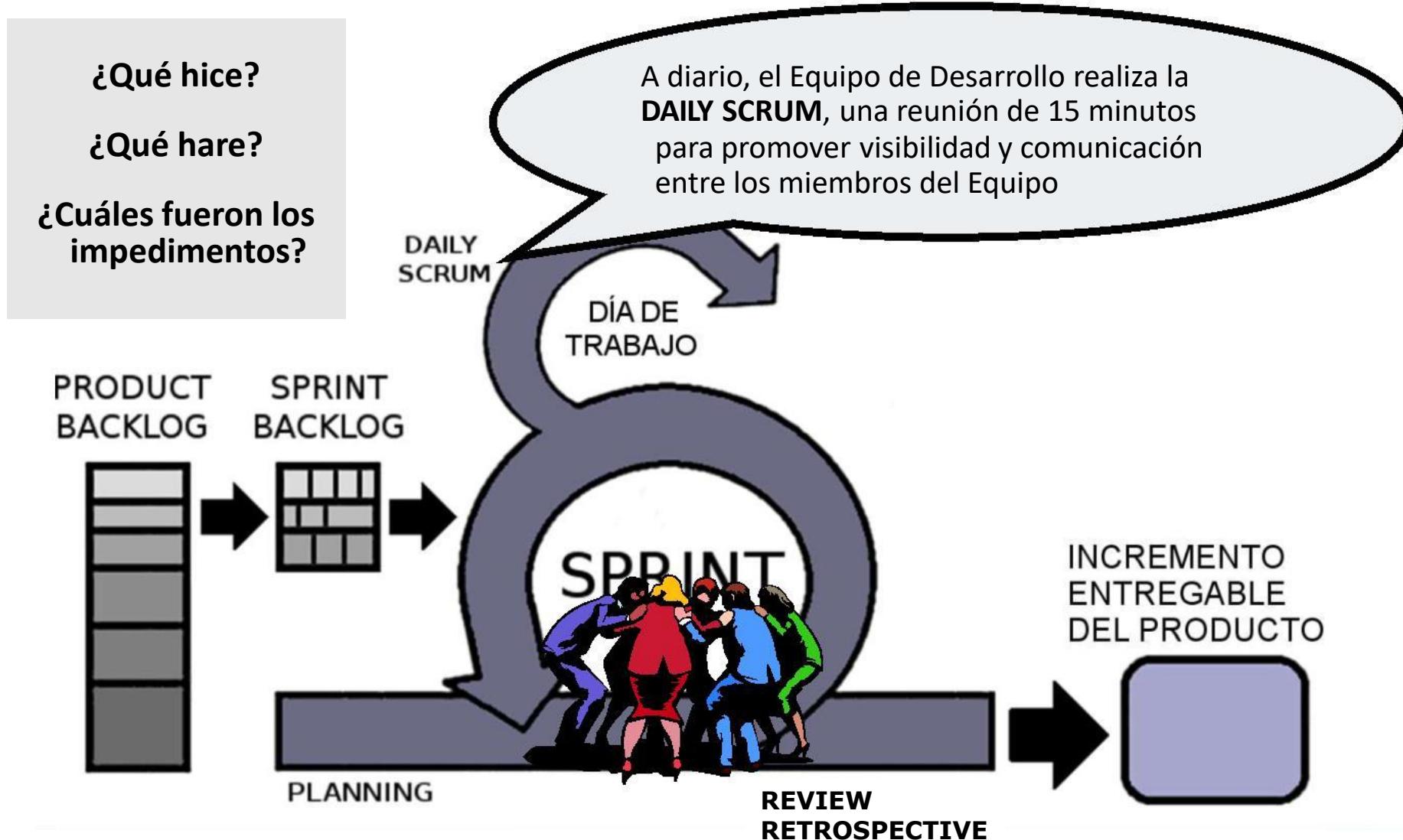
# Scrum: Sprint burndown



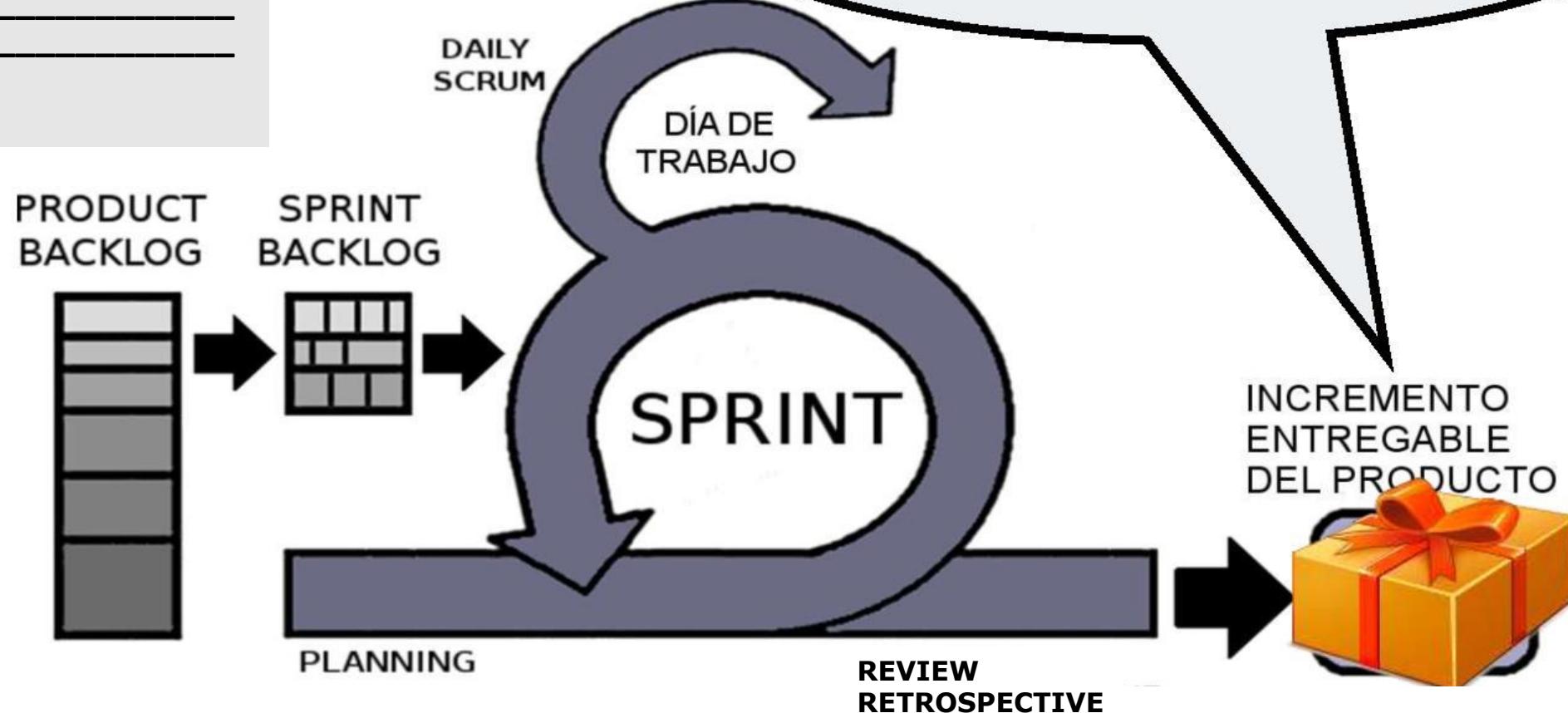
# Scrum: desarrollo del incremento



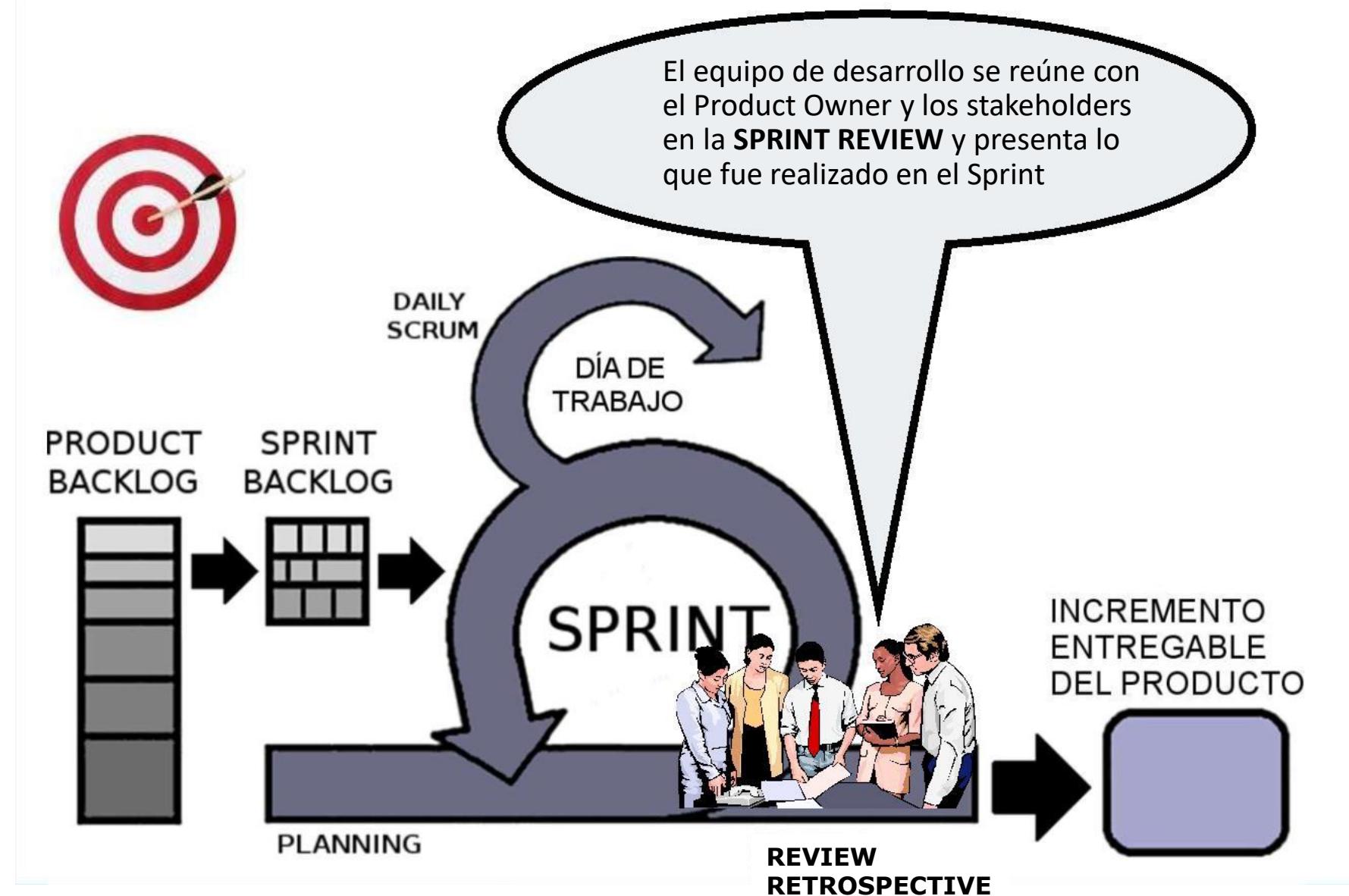
# Scrum: daily scrum



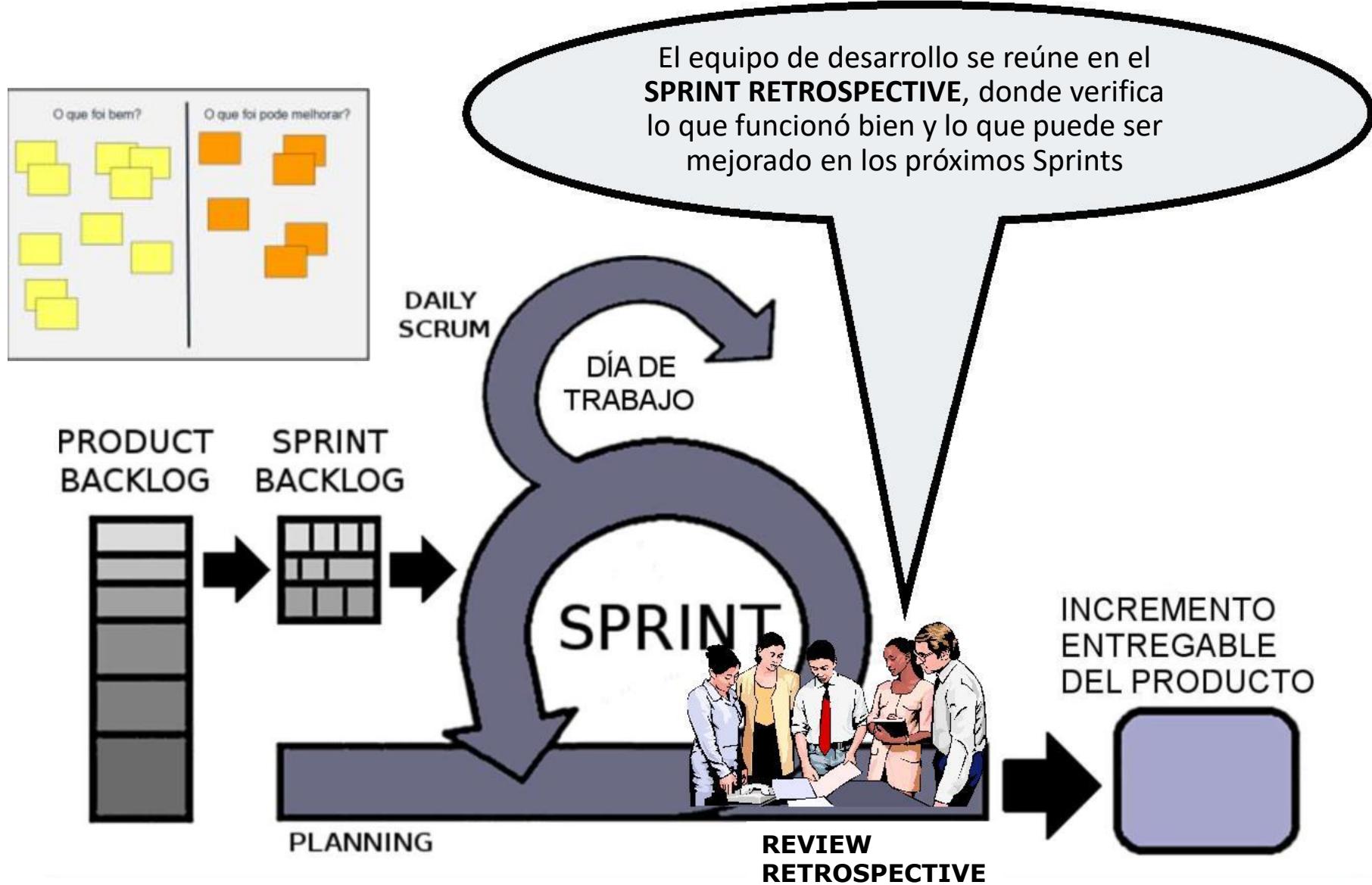
# Scrum: incremento del producto



# Scrum: sprint review

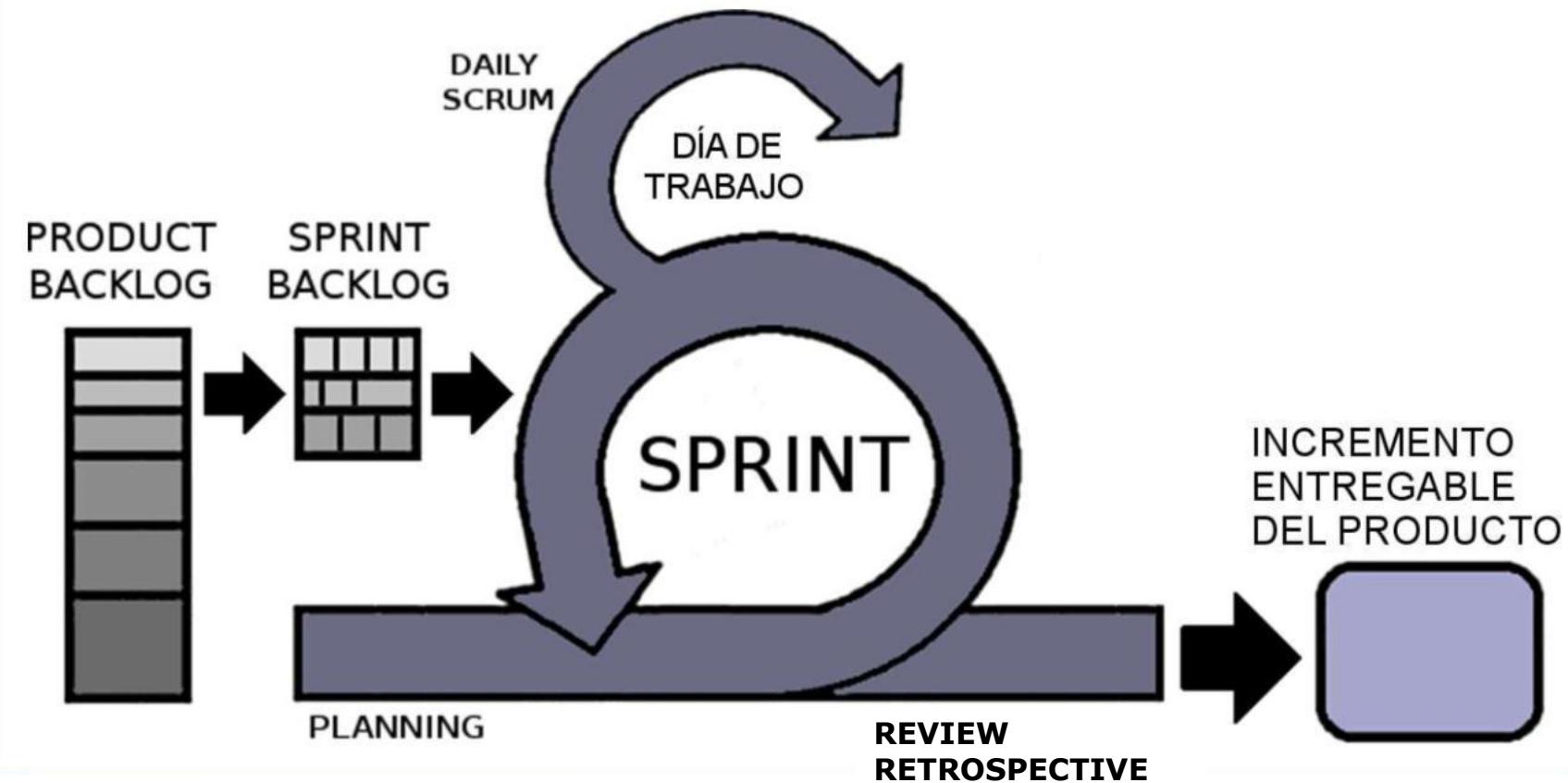


# Scrum: sprint retrospective



# Resumen del ciclo de Scrum

...y un nuevo ciclo comienza.



Ejemplo:

# Sistema de alquiler de vestidos

2m

4m

Release 1:

OFRECER CATALOGO DE VESTIDOS BASICO  
POR LA WEB

1. Mostrar vestidos

3. Separar vestido

5. Crear secciones por ocasión de uso

7. Realizar depósito de pago

9. Sección los más rentados

11. Solicitar vestido para una fecha

13. Mostrar vestidos recomendados

15. Generar categoría de clientes

2. Ver detalle Vestido

4. Acceder con FB

6. Filtro de vestido

8. Cargar vestidos por admin

10. Ofrecer beneficios por recomendar

12. Pagar en línea

14. Ofrecer descuentos

16. Generar historial de alquileres y puntos

Release 2:

CRECER Y MAXIMIZAR TICKET PROMEDIO

Release 3:

BENEFICIOS PARA LA ARRENDADORA

17. Inscripción y gestión de negocio

18. Control arriendos y beneficios

19. Depósito en línea

20. Sistema reconocimiento

Release 4 y 5:  
AUTOMATIZAR PROCESOS

21 Registro automático de nuevos vestidos

22 Seguimiento entrega vestido

23 Soporte y ayuda a clientes

24 Publicidad vía correo

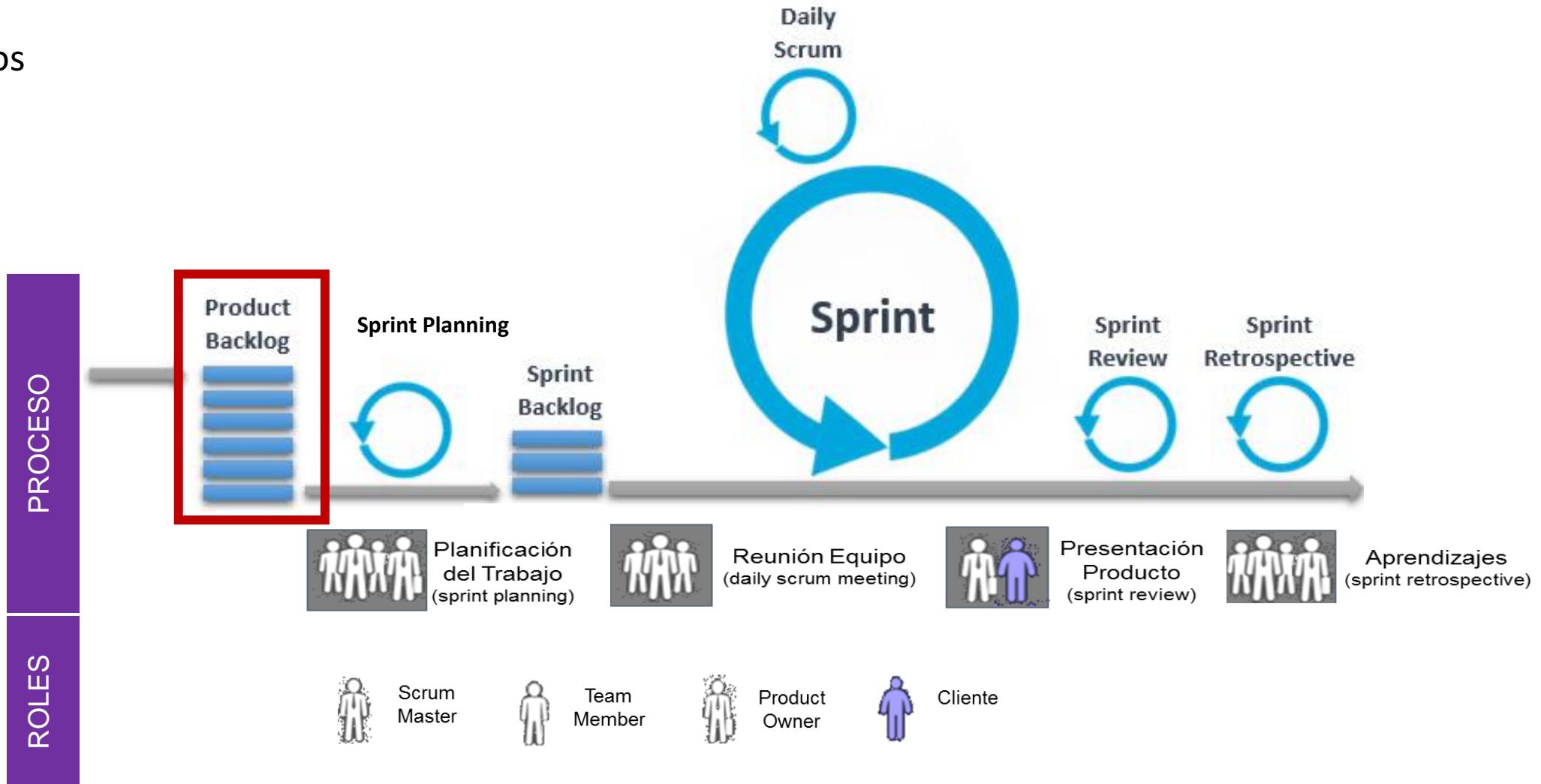
Release 6:

COMUNIDAD DE ALQUILER DE VESTIDOS

25 Desarrollo de una comunidad de alquiler de vestidos

# Scrum

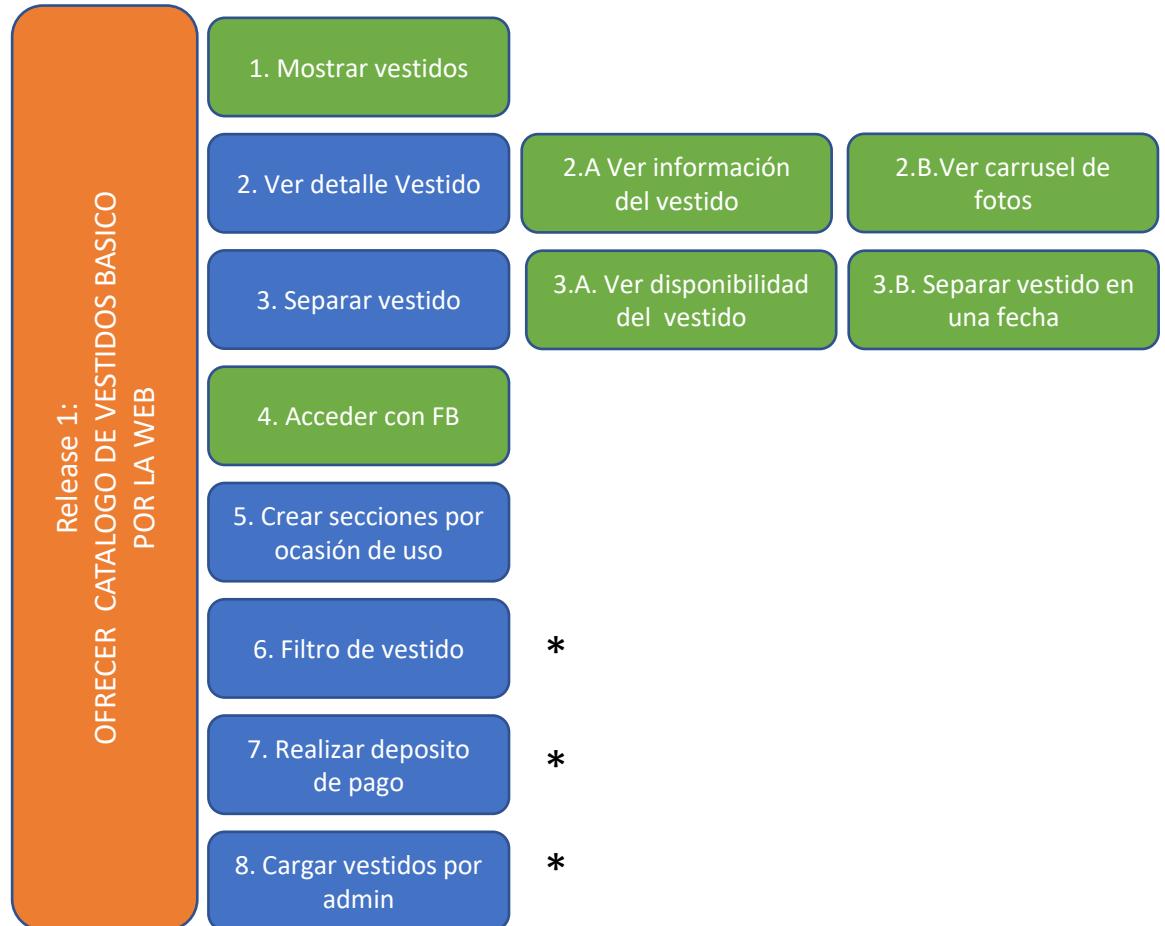
## Procesos



Ejemplo:

# Sistema de alquiler de vestidos

## GROOMING / REFINAMIENTO DE PRODUCT BACKLOG



\*: historias que aun se pueden dividir

Aquí verificamos que queda clara la historia.

**CARD:** Quede clara la necesidad

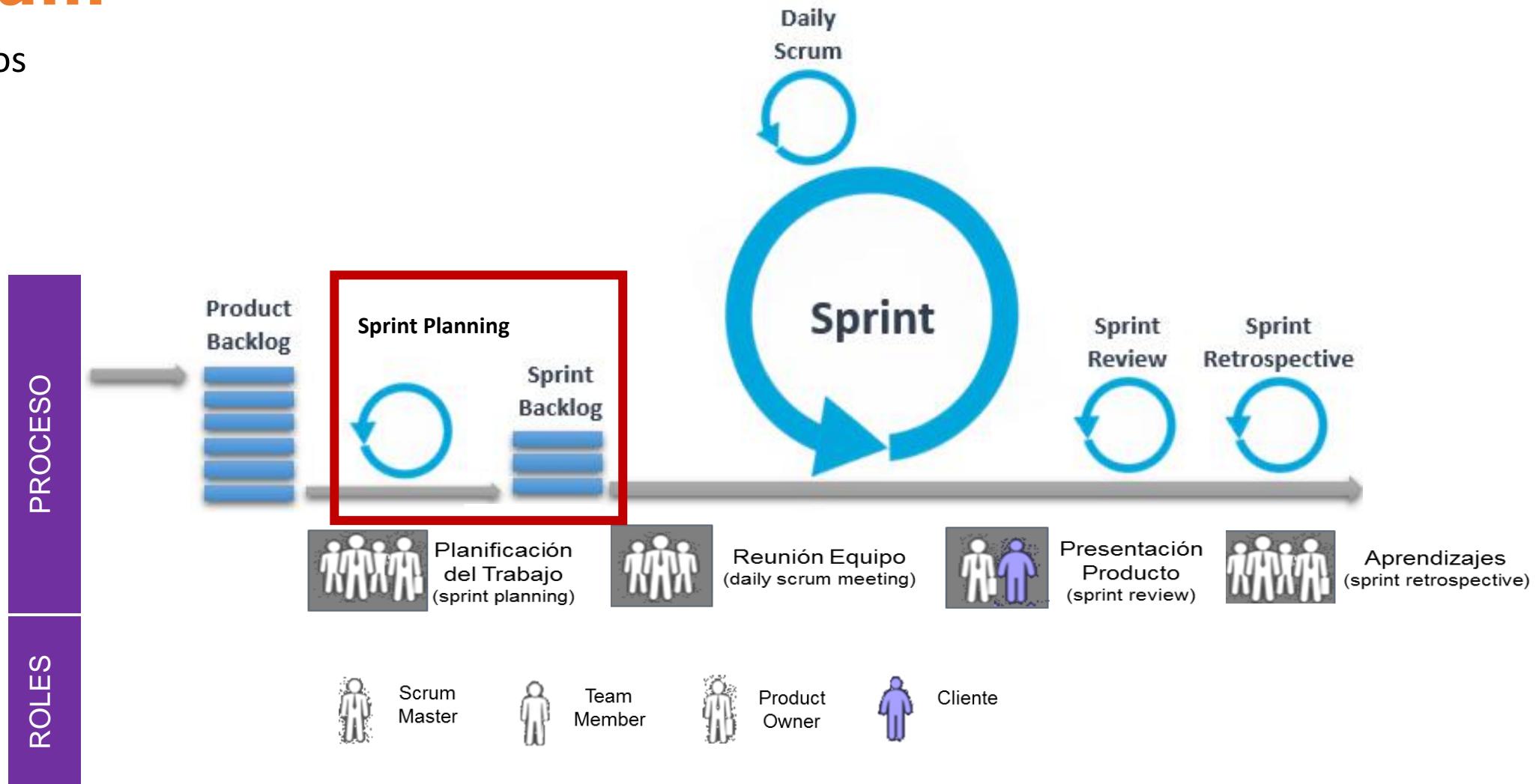
**CONVERSACION:** Queda claro los detalles y lineamientos para su elaboración. Para el caso de un desarrollo web es:

- Tener los diseños de pantallas,
- Lineamientos de arquitectura y
- Definiciones de integraciones

**CONFIRMACION:** Estén descritos los lineamientos de prueba, de manera específica

# Scrum

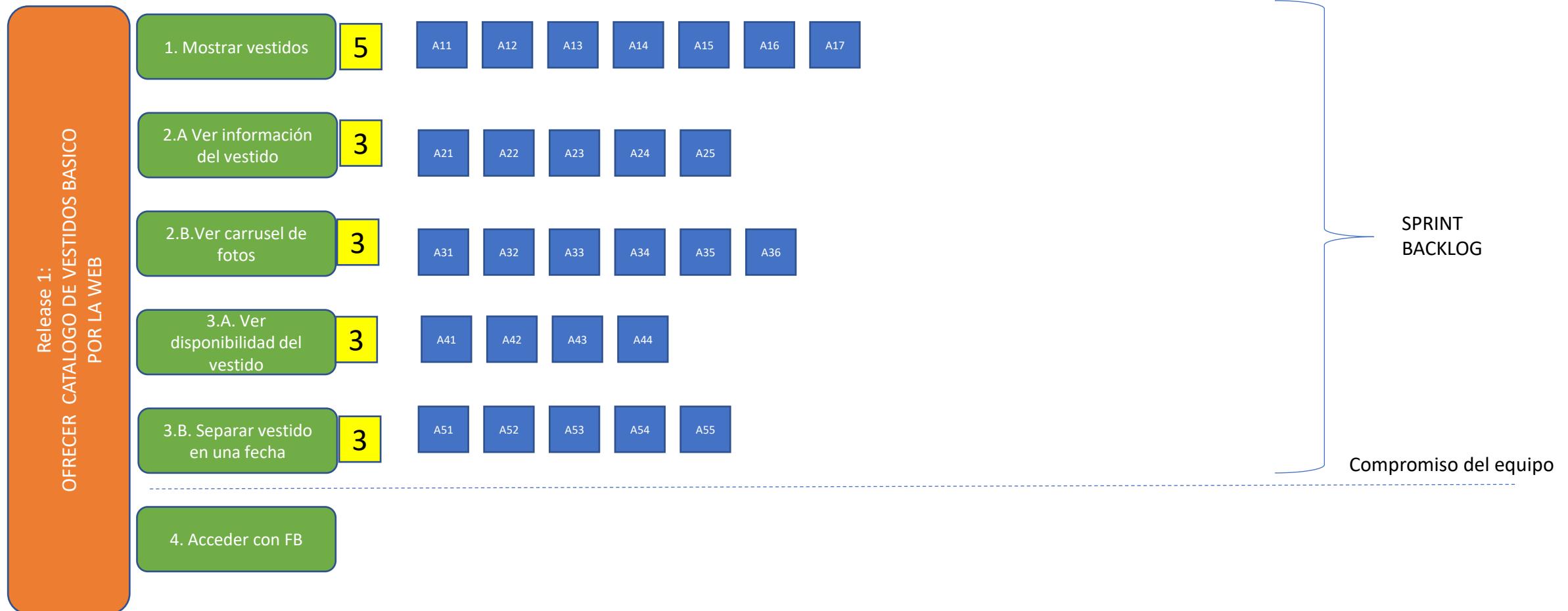
## Procesos



Ejemplo:

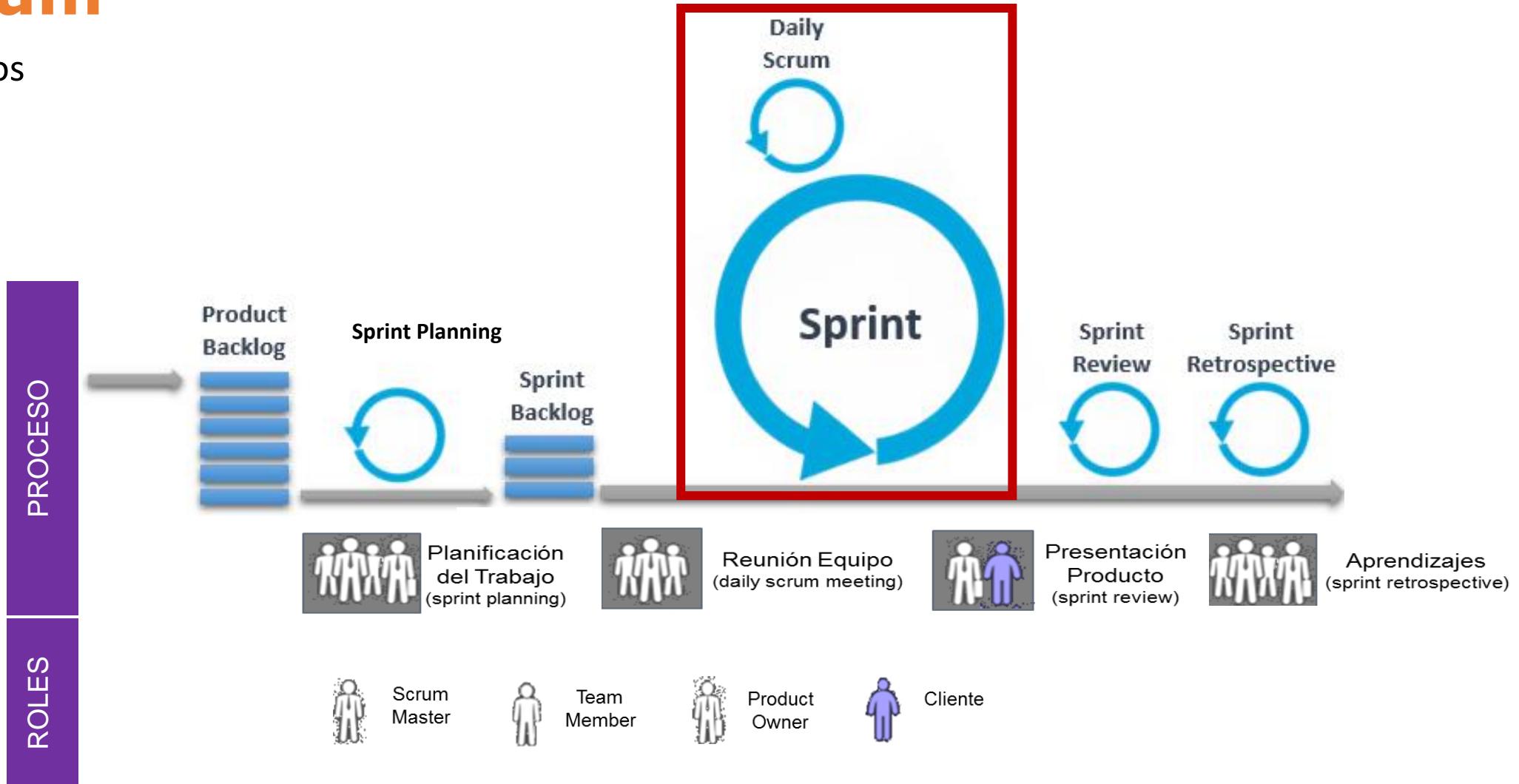
# Sistema de alquiler de vestidos

## SPRINT PLANNING



# Scrum

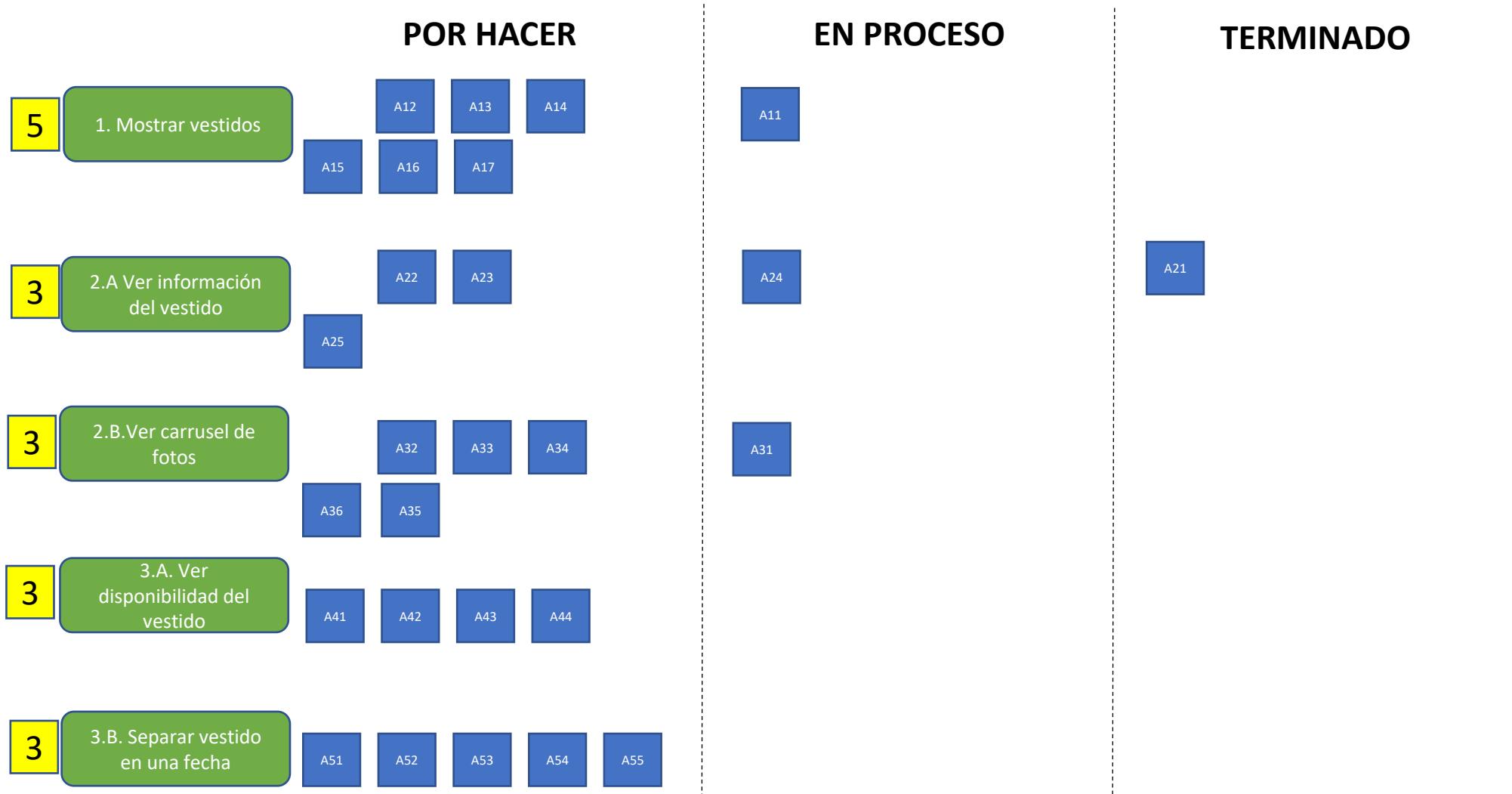
## Procesos



Ejemplo:

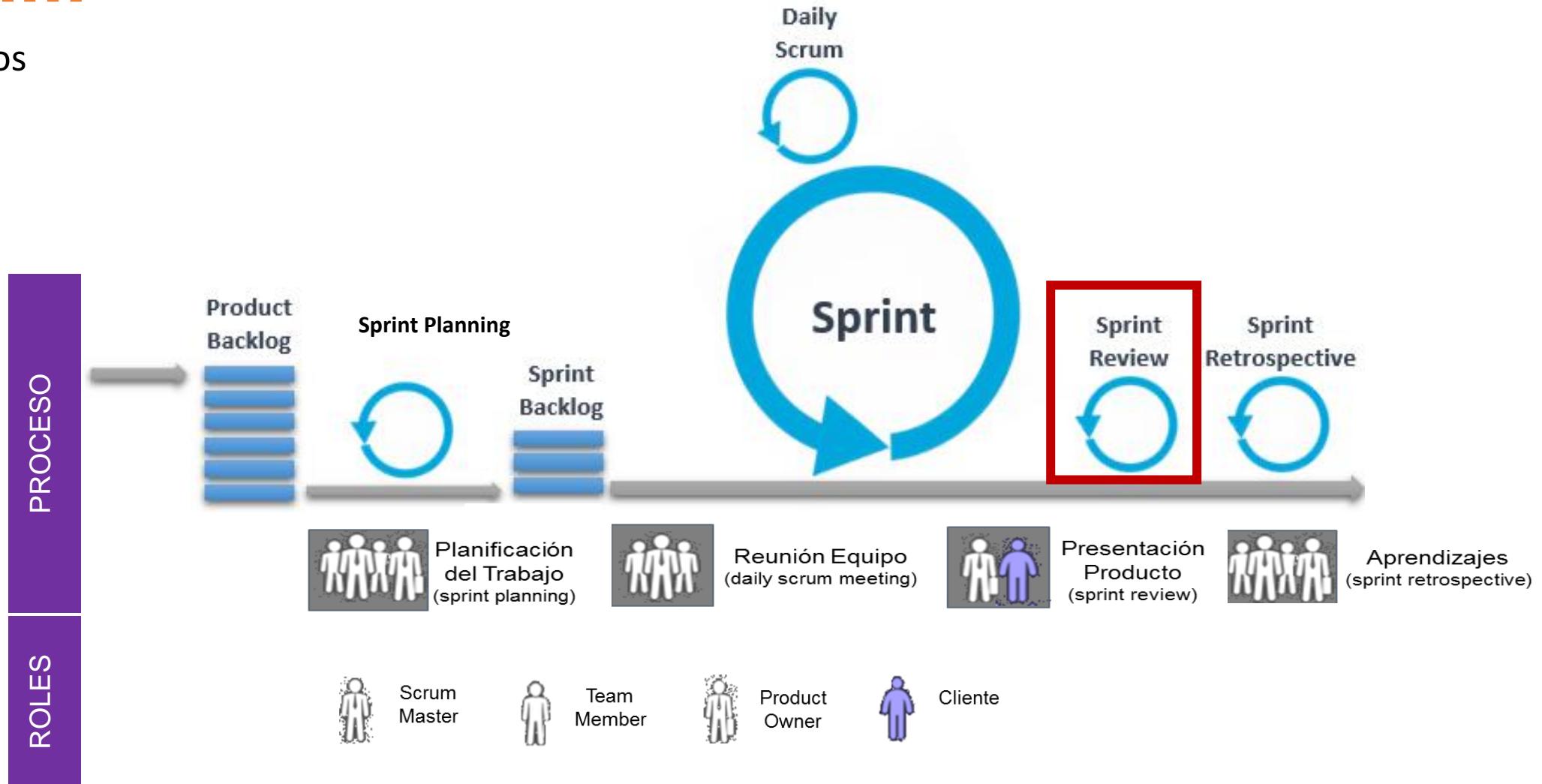
# Sistema de alquiler de vestidos

## DAILY SPRINT MEETING



# Scrum

## Procesos



Ejemplo:

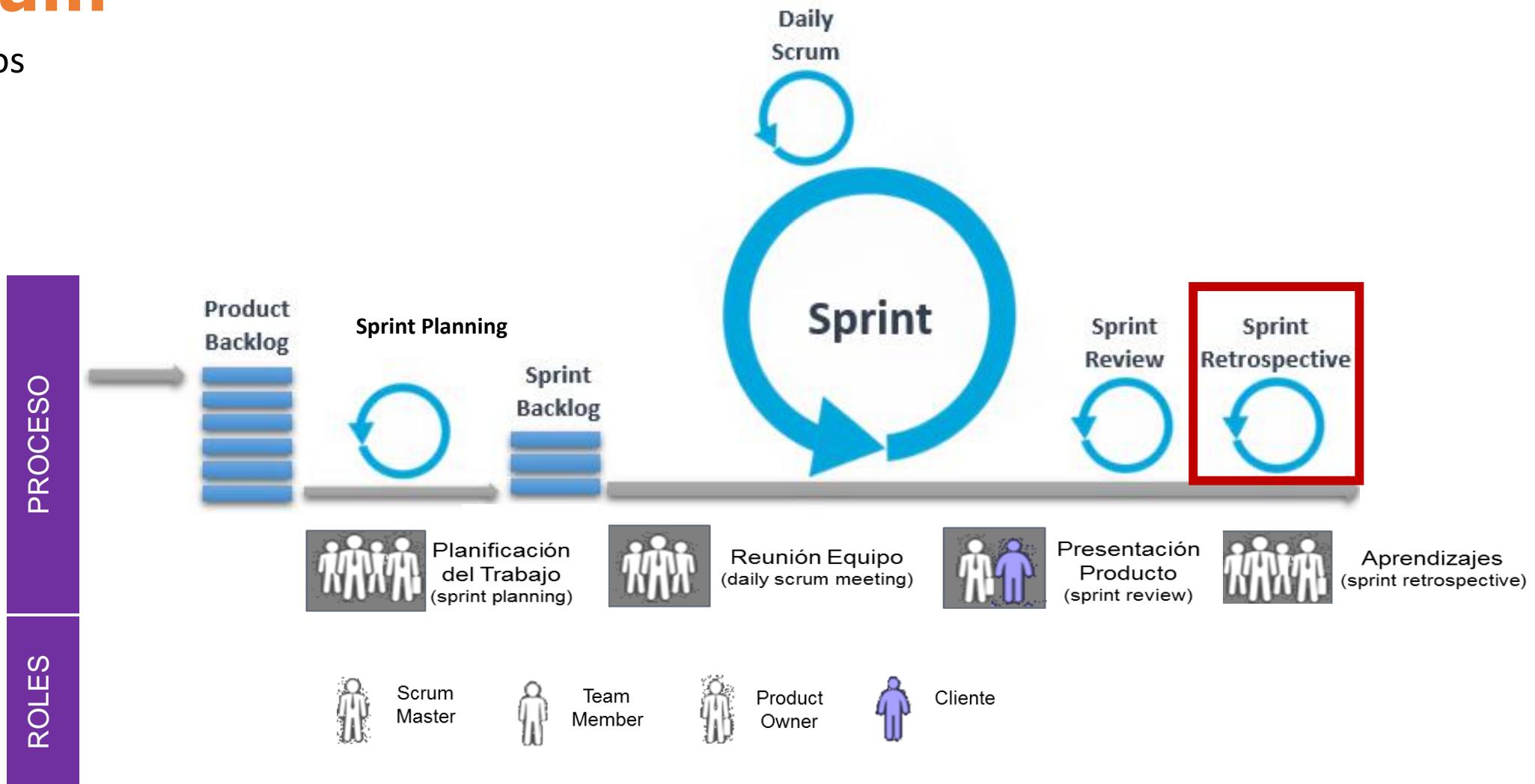
# Sistema de alquiler de vestidos

SPRINT REVIEW



# Scrum

## Procesos



Ejemplo:

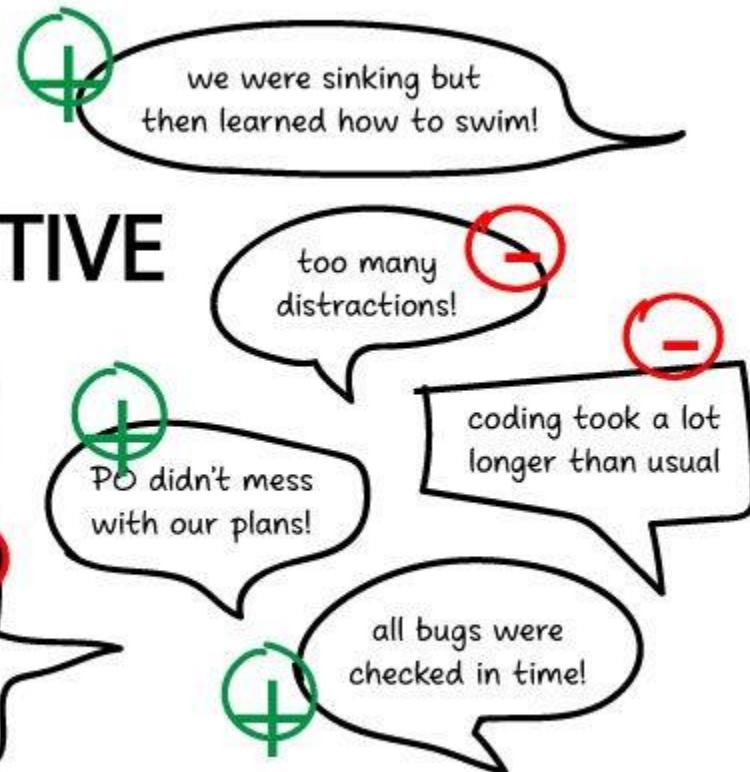
# Sistema de alquiler de vestidos

SPRINT RETROSPECTIVE



## RETROSPECTIVE

how do YOU  
think we did?





**UNIVERSIDAD NACIONAL DE INGENIERÍA**

**Facultad de Ingeniería  
Industrial y de Sistemas**

*Estándares de Ingeniería de Sistemas – SI-705  
Metodologías Agiles*